

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

1c986 U.S. PTO
09/784924
02/16/01

別紙添付の書類に記載されている事項は下記の出願書類に記載されて
いる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed
with this Office.

出 願 年 月 日

Date of Application:

2000年 2月21日

出 願 番 号

Application Number:

特願2000-043390

CERTIFIED COPY OF
PRIORITY DOCUMENT

出 願 人
Applicant(s):

ソニー株式会社

2001年 1月 5日

特許庁長官
Commissioner,
Patent Office

及 川 耕 造

出証番号 出証特2000-3109397

【書類名】 特許願

【整理番号】 00000202

【提出日】 平成12年 2月21日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 15/00

【発明の名称】 データ通信方法及びデータ通信装置、並びにプログラム
記憶媒体

【請求項の数】 23

【発明者】

 【住所又は居所】 東京都品川区北品川 6 丁目 7 番 3 5 号 ソニー株式会社
 内

 【氏名】 蒲生 勉

【特許出願人】

 【識別番号】 000002185

 【氏名又は名称】 ソニー株式会社

 【代表者】 出井 伸之

【代理人】

 【識別番号】 100101801

 【弁理士】

 【氏名又は名称】 山田 英治

 【電話番号】 03-5541-7577

【選任した代理人】

 【識別番号】 100093241

 【弁理士】

 【氏名又は名称】 宮田 正昭

 【電話番号】 03-5541-7577

【選任した代理人】

 【識別番号】 100086531

 【弁理士】

【氏名又は名称】 澤田 俊夫

【電話番号】 03-5541-7577

【手数料の表示】

【予納台帳番号】 062721

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9904833

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 データ通信方法及びデータ通信装置、並びにプログラム記憶媒体

【特許請求の範囲】

【請求項 1】

メッセージ通信を行う複数のオブジェクトからなるシステム環境下において、実行逐次性のある複数のオブジェクトで構成される複合オブジェクトと該複合オブジェクト外の独立オブジェクトとの間でメッセージ交換を行うためのデータ通信方法であって、

(a) 該複合オブジェクト内のオブジェクトによる該複合オブジェクト外の独立オブジェクトに対するメッセージ通信を一時蓄積するステップと、

(b) 該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したことに応答して、蓄積されたメッセージを該独立オブジェクトに対して一括的に送信するステップと、

を具備することを特徴とするデータ通信方法。

【請求項 2】

前記複合オブジェクトは、コンテキスト・スイッチが発生しない関数呼び出し形式で呼び出し可能な複数のオブジェクトで構成されることを特徴とする請求項 1 に記載のデータ通信方法。

【請求項 3】

前記複合オブジェクト内の各オブジェクトは、以下の制約条件の少なくとも 1 つを満たすことを特徴とする請求項 1 に記載のデータ通信方法。

(1) 各オブジェクト間に実行逐次性があることこと。

(2) 一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、メッセージ受信側におけるオブジェクトのスケジューリング実行スレッドの優先度が、メッセージ送信側におけるオブジェクトのそれと同等又はそれ以上であること。

(3) 一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、受信側におけるオブジェクトの実行スレッドの割り込み優先度が、送信側にお

けるオブジェクトのそれと同等か又はそれ以上であること。

(4) 各オブジェクトが同じメモリ保護属性を持つこと。

【請求項 4】

さらに、

(c) 該複合オブジェクト内におけるオブジェクトによるメッセージ通信履歴を作成するステップ、

を具備し、

前記ステップ (b) では、該メッセージ通信履歴を基に該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したか否かを判別する、

ことを特徴とする請求項 1 に記載のデータ通信方法。

【請求項 5】

さらに、

(c) 該複合オブジェクト内におけるオブジェクトによるメッセージ通信履歴を作成するステップ、

を具備し、

前記ステップ (b) では、前記複合オブジェクト内のオブジェクト実行終了時に該メッセージ通信履歴が異なる実行スレッドからの通信状態を示している場合に、蓄積されたメッセージを 1 回のメッセージ通信で送信先オブジェクトに送信する、

ことを特徴とする請求項 1 に記載のデータ通信方法。

【請求項 6】

前記ステップ (a) では、該複合オブジェクトと該独立オブジェクトの状態の関係に応じてメッセージ通信の蓄積を制御することを特徴とする請求項 1 に記載のデータ通信方法。

【請求項 7】

前記ステップ (a) では、該独立オブジェクトの状態に応じてメッセージ通信の蓄積を制御することを特徴とする請求項 1 に記載のデータ通信方法。

【請求項 8】

該複合オブジェクト内のオブジェクトのメッセージ送信先となる該複合オブジ

ェクト外の独立オブジェクトが複数存在する場合、前記ステップ（a）では、メッセージ送信先毎にメッセージ通信の蓄積を制御することを特徴とする請求項1に記載のデータ通信方法。

【請求項9】

さらに、

（d）送信元となる複合オブジェクト内のオブジェクトと送信先となる外部オブジェクトそれぞれの実行スレッドのスケジューリング優先度と割り込み優先度の関係に応じて、メッセージの蓄積を行うか又は即座に送信するかを決定するステップ、

を具備することを特徴とする請求項1に記載のデータ通信方法。

【請求項10】

前記複数のオブジェクトからなるシステムは、複数の並行オブジェクトで構成されたオブジェクト指向オペレーティング・システムであることを特徴とする請求項1に記載のデータ通信方法。

【請求項11】

前記複数のオブジェクトからなるシステムは、複数の並行オブジェクトで構成されたアプリケーション又はデバイス・ドライバであることを特徴とする請求項1に記載のデータ通信方法。

【請求項12】

メッセージ通信を行う複数のオブジェクトからなるシステム環境下において、実行逐次性のある複数のオブジェクトで構成される複合オブジェクトと該複合オブジェクト外の独立オブジェクトとの間でメッセージ交換を行うためのデータ通信装置であって、

（a）該複合オブジェクト内のオブジェクトによる該複合オブジェクト外の独立オブジェクトに対するメッセージ通信を一時蓄積する手段と、

（b）該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したことに応答して、蓄積されたメッセージを該独立オブジェクトに対して一括的に送信する手段と、

を具備することを特徴とするデータ通信装置。

【請求項 1 3】

前記複合オブジェクトは、コンテキスト・スイッチが発生しない関数呼び出し形式で呼び出し可能な複数のオブジェクトで構成されることを特徴とする請求項 1 2 に記載のデータ通信装置。

【請求項 1 4】

前記複合オブジェクト内の各オブジェクトは、以下の制約条件の少なくとも 1 つを満たすことを特徴とする請求項 1 2 に記載のデータ通信装置。

(1) 各オブジェクト間に実行逐次性があることこと。

(2) 一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、メッセージ受信側におけるオブジェクトのスケジューリング実行スレッドの優先度が、メッセージ送信側におけるオブジェクトのそれと同等又はそれ以上であること。

(3) 一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、受信側におけるオブジェクトの実行スレッドの割り込み優先度が、送信側におけるオブジェクトのそれと同等か又はそれ以上であること。

(4) 各オブジェクトが同じメモリ保護属性を持つこと。

【請求項 1 5】

さらに、

(c) 該複合オブジェクト内におけるオブジェクトによるメッセージ通信履歴を作成する手段、

を具備し、

前記手段 (b) は、該メッセージ通信履歴を基に該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したか否かを判別する、

ことを特徴とする請求項 1 2 に記載のデータ通信装置。

【請求項 1 6】

さらに、

(c) 該複合オブジェクト内におけるオブジェクトによるメッセージ通信履歴を作成する手段、

を具備し、

前記手段（b）は、前記複合オブジェクト内のオブジェクト実行終了時に該メッセージ通信履歴が異なる実行スレッドからの通信状態を示している場合に、蓄積されたメッセージを1回のメッセージ通信で送信先オブジェクトに送信する、ことを特徴とする請求項12に記載のデータ通信装置。

【請求項17】

前記手段（a）は、該複合オブジェクトと該独立オブジェクトの状態の関係に応じてメッセージ通信の蓄積を制御することを特徴とする請求項12に記載のデータ通信装置。

【請求項18】

前記手段（a）では、該独立オブジェクトの状態に応じてメッセージ通信の蓄積を制御することを特徴とする請求項12に記載のデータ通信装置。

【請求項19】

該複合オブジェクト内のオブジェクトのメッセージ送信先となる該複合オブジェクト外の独立オブジェクトが複数存在する場合、前記手段（a）では、メッセージ送信先毎にメッセージ通信の蓄積を制御することを特徴とする請求項12に記載のデータ通信装置。

【請求項20】

さらに、

（d）送信元となる複合オブジェクト内のオブジェクトと送信先となる外部オブジェクトそれぞれの実行スレッドのスケジューリング優先度と割り込み優先度の関係に応じて、メッセージの蓄積を行うか又は即座に送信するかを決定する手段を具備することを特徴とする請求項12に記載のデータ通信装置。

【請求項21】

前記複数のオブジェクトからなるシステムは、複数の並行オブジェクトで構成されたオブジェクト指向オペレーティング・システムであることを特徴とする請求項12に記載のデータ通信装置。

【請求項22】

前記複数のオブジェクトからなるシステムは、複数の並行オブジェクトで構成

されたアプリケーション又はデバイス・ドライバであることを特徴とする請求項 1 2 に記載のデータ通信装置。

【請求項 2 3】

メッセージ通信を行う複数のオブジェクトからなるシステム環境下において、実行逐次性のある複数のオブジェクトで構成される複合オブジェクトと該複合オブジェクト外の独立オブジェクトとの間でメッセージ交換を行うためのデータ通信処理をコンピュータ・システム上で実行するためのコンピュータ・プログラムをコンピュータ可読形式で記憶するプログラム記憶媒体であって、前記コンピュータ・プログラムは、

(a) 該複合オブジェクト内のオブジェクトによる該複合オブジェクト外の独立オブジェクトに対するメッセージ通信を一時蓄積するステップと、

(b) 該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したことに応答して、蓄積されたメッセージを該独立オブジェクトに対して一括的に送信するステップと、

を具備することを特徴とするプログラム記憶媒体。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、ソフトウェア・プログラム間におけるデータ通信に係り、特に、複数のオブジェクト間で行われるデータ通信に関する。

【0 0 0 2】

更に詳しくは、本発明は、複数の並列オブジェクトで構成されるオブジェクト指向オペレーティング・システムにおいてオブジェクト間で行われるデータ通信に係り、特に、コンテキスト・スイッチなどのコスト発生を可能な限り抑制したオブジェクト間の高速なデータ通信に関する。

【0 0 0 3】

【従来の技術】

昨今の L S I 技術の革新に伴い、情報処理機器、情報通信機器、家電機器など各種の機器は電子化され、マイクロプロセッサ上でソフトウェアを実行すること

によって機器動作の制御を行うことが一般的となってきた。

【0004】

また、旧来は、制御コードを始めとする各種ソフトウェア・プログラムは、機器に内蔵されるROM (Read Only Memory) 上に恒久的に書き込んでおくという形態で、出荷前にあらかじめ機器内に導入することが一般的であった。しかし、最近では、機器内に書き込み可能な記憶装置が装備され、各種記憶媒体を介して最新のソフトウェアを機器の出荷後に導入することも可能となってきた。さらに、多くの機器は、LANやインターネット、あるいは一般公衆回線などの通信媒体に接続されており、記憶媒体の物理的な流通に頼ることなく、最新のソフトウェアにリモート・アクセスすることが可能となっている。

【0005】

機器制御用のソフトウェアは、通常、オペレーティング・システム (OS) と呼ばれる形式で提供される。OSに関する厳格な定義はないが、機器ハードウェアと上位アプリケーションとの仲立ち (すなわち、アプリケーションに対するAPI (Application Programming Interface) の提供) と、機器ハードウェアの統括的な制御を、その基本的な役割とする。機器とOSとの組み合わせは、一般に「プラットフォーム」と呼ばれる。オペレーティング・システムは、アプリケーションやデバイス・ドライバに対してその実行環境を提供する。また、OSの介在により、異なる機種間で同一のアプリケーションを利用することが可能となる。

【0006】

ところで、ソフトウェア開発の分野では、処理手順ではなく処理対象となるデータを重要視するという「オブジェクト指向」 (object oriented) 技術が盛んに採り入れられている。一般に、オブジェクト指向によれば、ソフトウェアの開発と保守が効率化されと考えられている。オブジェクト指向に基づくソフトウェアは、基本的に、データとそのデータに対する処理手続きとを一体化させた「オブジェクト」というモジュール単位で扱われる。また、必要に応じて複数のオブジェクトを作成したり組み合わせることで1つのソフトウェアが完成する。

【0007】

オブジェクト指向パラダイムは、「カプセル化」、「クラス／インスタンス」、「クラス継承（インヘリタンス）」、及び、「メッセージ・パッシング」という4つの基本技術によって実現される。

【0008】

カプセル化は、データと手続き（メソッド）を一体化させることを指す。クラスは、複数のオブジェクトを共通化して定義することを指す。インスタンスはクラスに属するオブジェクトの実体を表す（同一クラスに属するオブジェクトすなわちインスタンスは基本的に同一のメソッドを有するので、個々にメソッドを定義する必要がない）。インヘリタンスとは、あるクラスで定義したことを別の（例えば下位の）クラスが継承することを意味する（新たにクラスを定義するときには、定義済みのクラスとの差分のみを追加変更するだけでよい）。メッセージ・パッシングとは、オブジェクトにメッセージを送って所定の動作を指示することを言う。各オブジェクトは自身のデータを隠蔽しているので、通常、メッセージ・パッシング以外の方法でオブジェクトにアクセスすることはできない。

【0009】

最近では、オペレーティング・システムに対しても、オブジェクト指向技術の適用が行われるようになってきている。すなわち、オペレーティング・システムの個々の構成要素を、実行主体である並行オブジェクトとしてモジュール化する。本明細書中では、以下、このようなオペレーティング・システムのことを「オブジェクト指向オペレーティング・システム」と呼び、また、並行オブジェクトのことを単に「オブジェクト」と呼ぶことにする。なお、オペレーティング・システムの構成要素となるオブジェクトと、オペレーティング・システム上で実行されるアプリケーションのオブジェクトが同様の実行機能を持つようなオブジェクト指向オペレーティング・システムのことを、特に、「純オブジェクト指向オペレーティング・システム」と称することもある。

【0010】

オブジェクト指向オペレーティング・システムでは、オペレーティング・システムが提供する各々のサービスはオブジェクトの集合として定義される。この特性を活かす帰結として、オブジェクト指向オペレーティング・システムは、システ

ム構成の柔軟性や、システムの動的変更の容易性などの点で、旧来のオペレーティング・システムよりも優れた性能を確保することができる。

【0011】

例えば、あるサービスを提供するためのオブジェクトを、オペレーティング・システムの実行開始前に適切に組み合わせることによって、それぞれのユーザにとって必要な機能を備えたオペレーティング・システムを容易に構築することができる。また、システムの最適化やアップグレードなどのためになされる機能追加や削除を、システム自体を停止させることなく動的に実施することも可能である。

【0012】

オブジェクト指向オペレーティング・システムでは、システムのサービスを提供する各オブジェクトが互いに並行して動作する。各オブジェクトはモジュール化されているので、メッセージ通信機構を用いることで、オブジェクト同士で通信してメッセージの交換を行ったり、動作の同期をとることができる。

【0013】

オブジェクト間のメッセージ通信には、オブジェクトの実行スレッドの状態の待避とその復元に伴うコスト、メッセージの配送のコストなどを費やしてしまう。このため、オブジェクト指向オペレーティング・システムの場合、オブジェクト間のメッセージ通信が頻繁に生じると、その回数だけコンテキスト・スイッチが発生してしまうので、システム全体の性能が劣化してしまうという問題がある。

#d

したがって、オブジェクト指向オペレーティング・システムのシステム設計者においては、システム構成の柔軟性やシステムの動的変更の容易性などオブジェクト指向オペレーティング・システムの優れた特徴を採り入れるだけでなく、かかる特徴と実際の実行性能とのバランスを十分に考慮した上でオブジェクトを設計する必要がある。

【0014】

例えば、本出願人に既に譲渡されている特願平11-57689号明細書には

、オブジェクト間のメッセージ通信時のコストを削減することでシステム全体の実行性能を向上させることができるデータ処理方法について開示されている。同明細書に記載のデータ処理方法では、メッセージ通信が頻繁に発生するような複数のオブジェクトを包括するようなオブジェクト、すなわち「複合オブジェクト」を定義するようになっている。

【0015】

複合オブジェクトを構成する各オブジェクトのプログラミング・スタイルは、複合オブジェクト外の通常の並行オブジェクトのそれと同様であり（すなわち変更を要しない）、オブジェクト識別子（Object Identifier: O I D）を有して、通常のオブジェクトからも参照可能である。また、複合オブジェクト内の各オブジェクトは、独自の実行スレッドを持たず、複合オブジェクトが持つ実行スレッドを共有するようになっている。そして、同一の複合オブジェクト内でオブジェクト間のメッセージ通信を行う時には、実行スレッドの切り替えすなわちコンテキスト・スイッチが発生しないので、メッセージ通信のコストを関数呼び出しと同等にすることができる。

【0016】

オブジェクト指向オペレーティング・システム上で複合オブジェクトを構成することにより、システムの柔軟性を維持したまま、メッセージ通信のコストを削減することができるので、同特許出願に係るデータ処理方法は非常に有効と言えよう。

【0017】

ここで、2以上のオブジェクトを包括して複合オブジェクトを構成する場合には、元の各オブジェクトに対しては所定の制約条件が課される。この制約条件は、オペレーティング・システムが提供する機能によって若干の相違はあるが、おおむね以下の通りである。

【0018】

（1）各オブジェクト間に実行逐次性があること。より具体的には、以下の2つの条件を満たす場合に「実行逐次性」が認められる。すなわち、

（1-1）一方のオブジェクトから他方のオブジェクトにメッセージを送信する

とき、両オブジェクトが並列動作する必要がない。

(1-2) 一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、メッセージ受信側のオブジェクトが他のメッセージを処理中であることがない。

【0019】

(2) 一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、メッセージ受信側におけるオブジェクトのスケジューリング実行スレッドの優先度が、メッセージ送信側におけるオブジェクトのそれと同等又はそれ以上であること。

【0020】

(3) 一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、受信側におけるオブジェクトの実行スレッドの割り込み優先度が、送信側におけるオブジェクトのそれと同等か又はそれ以上であること。

【0021】

(4) 各オブジェクトが同じメモリ保護属性を持つこと。

【0022】

言い換えれば、上記の制約条件を満足しないようなオブジェクト同士を構成オブジェクトとして複合オブジェクトを構成することはできない。特願平11-57689号明細書は、複合オブジェクトを構成するための手順について詳解している。但し、すべてのオブジェクトを複合オブジェクトとして包括できるものではない。

【0023】

このため、頻繁にメッセージ通信を行うような特定のオブジェクトの組み合わせがあったとしても、上記の制約条件を満足しない限り、複合オブジェクトに包括せず、通常の並行オブジェクトとして独立させる必要がある。このため、頻繁に行われるメッセージ通信のために、実行スレッドの切り替えすなわちコンテキスト・スイッチが発生してしまい、メッセージ通信のコストを削減することができない。

【0024】

例えば、タスクのスケジューリングを管理するスケジューラのようなオブジェクトは、特定の複合オブジェクトの内部に包括することはできない。したがって、複合オブジェクト内でメッセージ通信を行う場合であっても、各構成オブジェクトがスケジューラをコールする度にコンテキストが切り替わり、コストが増大してしまう。スケジューラのコールは、データ処理の過程において、極めて頻繁に発生する。

【0025】

【発明が解決しようとする課題】

本発明の目的は、複数のオブジェクト間でメッセージ交換を行うための、優れたデータ通信方式を提供することにある。

【0026】

本発明の更なる目的は、複数の並列オブジェクトで構成されるオブジェクト指向オペレーティング・システムにおいてオブジェクト間でのメッセージ交換を好適に行うことができるの、優れたデータ通信方式を提供することにある。

【0027】

本発明の更なる目的は、コンテキスト・スイッチなどのコスト発生を可能な限り抑制して、オブジェクト間のメッセージ交換を高速化することができる、優れたデータ通信方式を提供することにある。

【0028】

本発明の更なる目的は、オブジェクト指向オペレーティング・システムにおけるシステム構成の柔軟性や動的変更の容易性などの優位性を維持したままで、オブジェクト間のメッセージ通信時のコストを削減して、システム全体の実行性能を向上させることにある。

【0029】

本発明の更なる目的は、複合オブジェクト内の各構成オブジェクトが外部の特定のオブジェクトとの間で頻繁にメッセージ通信を行う場合であっても、各オブジェクトの独立性を保ちつつ、コンテキスト・スイッチに伴うコストを削減して、システム全体の実行性能を向上させることにある。

【0030】

【課題を解決するための手段及び作用】

本発明は、上記課題を参酌してなされたものであり、その第1の側面は、メッセージ通信を行う複数のオブジェクトからなるシステム環境下において、実行逐次性のある複数のオブジェクトで構成される複合オブジェクトと該複合オブジェクト外の独立オブジェクトとの間でメッセージ交換を行うためのデータ通信方法又はデータ通信装置であって、

(a) 該複合オブジェクト内のオブジェクトによる該複合オブジェクト外の独立オブジェクトに対するメッセージ通信を一時蓄積するステップ又は手段と、

(b) 該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したことに応答して一時蓄積されたメッセージを該独立オブジェクトに対して一時に送信するステップ又は手段と、

を具備することを特徴とするデータ通信方法又はデータ通信装置である。

【0031】

本発明の第1の側面に係るデータ通信方法又はデータ通信装置は、例えば、メッセージ通信を行う複数の並行オブジェクトからなるオブジェクト指向オペレーティング・システムが導入されたデータ処理システム上で実現される。

【0032】

該データ通信方法及びデータ通信装置によれば、まず、メッセージ通信が頻繁に発生するような複数のオブジェクトを包括した複合オブジェクトが生成される。複合オブジェクトは実行逐次型のオブジェクトで構成され、複合オブジェクト内でのオブジェクト間通信は関数呼び出しで実現するので、メッセージ通信に伴うコンテキスト・スイッチすなわちコストは発生しない。したがって、同一の複合オブジェクト内のオブジェクトがコールされたときには、同一の実行スレッドによってメッセージ通信を実行することができる。

【0033】

他方、複合オブジェクト内から外部の独立オブジェクトに対してメッセージ通信を行うと、コンテキストが切り替わる。複合オブジェクト内においてオブジェクト間通信を行う場合であっても、スケジューラなどの複合オブジェクト外に存在する独立オブジェクトを頻繁にコールしなければならないことがある。コール

の度にメッセージ通信を実行していると、コンテキスト・スイッチの発生によりコストが著しく増大してしまう。

【0034】

そこで、本発明の第1の側面に係るデータ通信方法及びデータ通信装置では、複合オブジェクト内のオブジェクトが外部の独立オブジェクトをコールしたときには、逐次的にメッセージ通信を実行する代わりに、独立オブジェクトへのメッセージ通信をキュー内に一時的に蓄積しておく。そして、複合オブジェクトと独立オブジェクトの関係が所定の状態に到達したときに、独立オブジェクトへのメッセージを一括して送信するようにした。

【0035】

この結果、コンテキスト・スイッチなどのコスト発生を可能な限り抑制して、オブジェクト間のメッセージ交換を高速化することができる。

【0036】

また、本発明によれば、オブジェクト指向オペレーティング・システムにおけるシステム構成の柔軟性や動的変更の容易性などの優位性を維持したままで、オブジェクト間のメッセージ通信時のコストを削減して、システム全体の実行性能を向上させることができる。

【0037】

また、本発明によれば、複合オブジェクト内の各構成オブジェクトが外部の特定のオブジェクトとの間で頻繁にメッセージ通信を行う場合であっても、各オブジェクトの独立性を保ちつつ、コンテキスト・スイッチに伴うコストを削減して、システム全体の実行性能を向上させることができる。

【0038】

ここで、複合オブジェクト内の各オブジェクトは、例えば以下に示すような制約条件を満足している。すなわち、

- (1) 各オブジェクト間に実行逐次性があること。
- (2) 一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、メッセージ受信側におけるオブジェクトのスケジューリング実行スレッドの優先度が、メッセージ送信側におけるオブジェクトのそれと同等又はそれ以上であ

ること。

(3) 一方のオブジェクトから他方のオブジェクトにメッセージを送信するとき、受信側におけるオブジェクトの実行スレッドの割り込み優先度が、送信側におけるオブジェクトのそれと同等か又はそれ以上であること。

(4) 各オブジェクトが同じメモリ保護属性を持つこと。

【0039】

また、本発明の第1の側面に係るデータ通信方法又はデータ通信装置は、さらに、(c) 該複合オブジェクト内におけるオブジェクトによるメッセージ通信履歴を作成するステップ又は手段を具備してもよい。このような場合、前記ステップ又は手段(b)では、該メッセージ通信履歴を基に該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したか否かを判別することができる。あるいは、前記ステップ又は手段(b)では、前記複合オブジェクト内のオブジェクト実行終了時に該メッセージ通信履歴が異なる実行スレッドからの通信状態を示している場合には、蓄積されたメッセージを1回のメッセージ通信で送信先オブジェクトに送信することができる。

【0040】

また、前記ステップ又は手段(a)では、該複合オブジェクトと該独立オブジェクトの状態の関係に応じてメッセージ通信の蓄積を制御するようにしてもよい。

【0041】

あるいは、前記ステップ又は手段(a)では、該独立オブジェクトの状態に応じてメッセージ通信の蓄積を制御するようにしてもよい。

【0042】

例えば、複合オブジェクト内でのオブジェクト間通信、すなわち複合オブジェクト内のあるオブジェクトによるメッセージ送信先が同一複合オブジェクト内の他のオブジェクトである場合には、同一の実行スレッドでメッセージ通信を実行する。このとき、メッセージ通信履歴情報を同一実行スレッドからの通信状態に更新すればよい。

【0043】

これに対し、複合オブジェクト内のオブジェクトのメッセージ送信先が複合オブジェクト外に存在する通常のオブジェクトである場合には、即座にメッセージ通信を行わずに、メッセージ送信先に応じてメッセージを一旦蓄積する。そして、複合オブジェクト内でのメッセージ処理が終了すると、メッセージ通信履歴情報を参照する。メッセージ通信履歴情報が同一実行スレッドからの通信状態を示しているときは復帰処理を行う。他方、メッセージ通信履歴が異なる実行スレッドからの通信状態を示しているときには、蓄積されたメッセージを1回のメッセージ通信で送信先オブジェクトに送信して、複合オブジェクトとしてのメソッドの実行を完了させる。

【0044】

また、該複合オブジェクト内のオブジェクトのメッセージ送信先となる該複合オブジェクト外の独立オブジェクトが複数存在する場合、前記ステップ又は手段(a)では、メッセージ送信先毎にメッセージ通信の蓄積を制御するようにしてもよい。

【0045】

また、送信元となる複合オブジェクト内のオブジェクトと送信先となる外部オブジェクトそれぞれの実行スレッドのスケジューリング優先度と割り込み優先度の関係に応じて、メッセージの蓄積を行うか又は即座に送信するかを決定するようにしてもよい。

【0046】

本発明の第1の側面に係るデータ通信方法によれば、複合オブジェクト内のオブジェクトと複合オブジェクト外に存在するオブジェクトとのメッセージ通信が頻繁に行われている場合、複合オブジェクト内の各オブジェクトには透過的な形式でメッセージが蓄積され、その後の適切なタイミングで蓄積された複数のメッセージをまとめて1回のメッセージ通信による送信することができる。この結果、メッセージ通信のコストを大幅に削減することができる。また、オブジェクトの独立性を保持したままシステム全体の実行性能を向上させることができる。

【0047】

本発明の第1の側面に係るデータ通信方法は、オブジェクト指向オペレーティ

ング・システムだけではなく、複数の並行オブジェクトで構成されたアプリケーションやデバイス・ドライバに対しても適用することができる。

【 0 0 4 8 】

また、本発明の第 2 の側面は、メッセージ通信を行う複数のオブジェクトからなるシステム環境下において、実行逐次性のある複数のオブジェクトで構成される複合オブジェクトと該複合オブジェクト外の独立オブジェクトとの間でメッセージ交換を行うためのデータ通信処理をコンピュータ・システム上で実行するためのコンピュータ・プログラムをコンピュータ可読形式で記憶するプログラム記憶媒体であって、前記コンピュータ・プログラムは、

(a) 該複合オブジェクト内のオブジェクトによる該複合オブジェクト外の独立オブジェクトに対するメッセージ通信を一時蓄積するステップと、

(b) 該複合オブジェクトと該独立オブジェクトの状態が所定の関係に到達したことに応答して、蓄積されたメッセージを該独立オブジェクトに対して一括的に送信するステップと、

を具備することを特徴とするプログラム記憶媒体である。

【 0 0 4 9 】

本発明の第 2 の側面に係るコンピュータ可読記憶媒体は、例えば、様々なプログラム・コードを実行可能な汎用コンピュータ・システムに対して、コンピュータ・プログラムをコンピュータ可読な形式で提供する媒体である。このような媒体は、例えば、CD (Compact Disc) や FD (Floppy Disc)、MO (Magneto-Optical disc) などの着脱自在で可搬性の記憶媒体である。あるいは、ネットワーク (ネットワークは無線、有線の区別を問わない) などの伝送媒体などを經由してコンピュータ・プログラムを特定のコンピュータ・システムに提供することも技術的に可能である。

【 0 0 5 0 】

このようなプログラム記憶媒体は、コンピュータ・システム上で所定のコンピュータ・プログラムの機能を実現するための、コンピュータ・プログラムと記憶媒体との構造上又は機能上の協働的關係を定義したものである。換言すれば、本発明の第 2 の側面に係るプログラム記憶媒体を介して所定のコンピュータ・プロ

グラムをコンピュータ・システムにインストールすることによって、コンピュータ・システム上では協働的作用が発揮され、本発明の第1の側面と同様の作用効果を得ることができる。

【0051】

本発明のさらに他の目的、特徴や利点は、後述する本発明の実施例や添付する図面に基づくより詳細な説明によって明らかになるであろう。

【0052】

【発明の実施の形態】

以下、図面を参照しながら本発明の実施例を詳解する。

【0053】

図1には、本発明の実施に供されるデータ処理システム10のハードウェア構成を模式的に示している。データ処理システム10の一例は、米IBM社のパーソナル・コンピュータPC/AT (Personal Computer/Advanced Technology) の互換機又は後継機を始めとする情報処理機器であるが、その他のタイプの情報処理機器や情報家電機器であってもよい。以下、システム10の各部について説明する。

【0054】

システム10のメイン・コントローラであるプロセッサ11は、例えば、CPU (Central Processing Unit) と呼ばれるLSI (Large Scale Integration) チップで構成され、オペレーティング・システムの制御下で、各種のアプリケーションを実行するようになっている。

【0055】

図示の通り、プロセッサ11は、バス17によって他の機器類と相互接続されている。バス17上の各機器にはそれぞれ固有のメモリ・アドレス又はI/Oアドレスが付与されており、プロセッサ11はこれらアドレスによって機器アクセスが可能となっている。バス17の一例はPCI (Peripheral Component Interconnect) バスである。

【0056】

メモリ12は、プロセッサ11において実行されるプログラム・コードを格納

したり、実行中の作業データを一時保管するために使用される記憶装置である。本実施例では、メモリ 1 2 は、不揮発及び揮発メモリ双方を含むものと理解されたい。

【 0 0 5 7 】

ディスプレイ・コントローラ 1 3 は、プロセッサ 1 1 が発行する描画命令を実際に処理するための専用コントローラであり、例えば S V G A (Super Video Graphic Array) 又は X G A (eXtended Graphic Array) 相当の描画機能をサポートする。ディスプレイ・コントローラ 1 3 において処理された描画データは、例えばフレーム・バッファ (図示しない) に一旦書き込まれた後、表示装置 2 1 に画面出力される。表示装置 2 1 は、例えば、C R T (Cathode Ray Tube) ディスプレイや、液晶表示ディスプレイ (Liquid Crystal Display) などである。

【 0 0 5 8 】

入力機器インターフェース 1 4 は、キーボード 2 2 やマウス 2 3 などのユーザ入力機器をシステム 1 0 に接続するための装置である。入力機器インターフェース 1 4 は、キーボード 2 2 によるキー入力又はマウス 2 3 を介した座標指示入力に応答して、プロセッサ 1 1 に対して割り込みを発生する。

【 0 0 5 9 】

ネットワーク・インターフェース 1 5 は、E t h e r n e t などの所定の通信プロトコルに従って、システム 1 0 を L A N (Local Area Network) などのネットワークに接続することができる。ネットワーク・インターフェース 1 5 は、一般に、L A N アダプタ・カードの形態で提供され、マザーボード (図示しない) 上の P C I バス・スロットの装着して用いられる。

【 0 0 6 0 】

L A N 上には、複数のホスト (コンピュータ) がトランスペアレントに接続され、分散コンピューティング環境が構築されている。また、ホストの一部はルータとして稼動し、さらに他の L A N やインターネットなどの外部ネットワークに接続されている。インターネット上では、ソフトウェア・プログラムやデータ・コンテンツなどのディストリビューションが行われる (周知) 。

【 0 0 6 1 】

HDD (Hard Disc Drive) インターフェース 1 6 は、ハード・ディスク・ドライブなどの外部記憶装置をシステム 1 0 に接続するための装置である。HDD インターフェース 1 6 は、例えば、I D E (Integrated Drive Electronics) や S C S I (Small Computer System Interface) などのインターフェース規格に準拠する。

【 0 0 6 2 】

HDD 2 4 は、記憶担体としての磁気ディスクを固定的に搭載した外部記憶装置であり（周知）、記憶容量やデータ転送速度などの点で他の外部記憶装置よりも優れている。ソフトウェア・プログラムを実行可能な状態で HDD 2 6 上に置くことをプログラムのシステムへの「インストール」と呼ぶ。通常、HDD 2 4 には、プロセッサ 1 1 が実行すべきオペレーティング・システムのプログラム・コードや、アプリケーション・プログラム、デバイス・ドライバなどが不揮発的に格納されている。

【 0 0 6 3 】

本実施例で言うオペレーティング・システムは、個々の構成要素が実行主体としての並列オブジェクトとしてモジュール化されて構成された「オブジェクト指向オペレーティング・システム」である。オブジェクト指向に基づくソフトウェアは、基本的に、データとそのデータに対する処理手続きとを一体化させた「オブジェクト」というモジュール単位で扱われる。オブジェクト指向プログラミングによれば、必要に応じて複数のオブジェクトを作成したり組み合わせることで 1 つのソフトウェアが完成する。

【 0 0 6 4 】

また、本実施例では、アプリケーションやデバイス・ドライバなどのオブジェクトで構成されるものとする。さらに、アプリケーション用の実行環境（以下、本明細書では“mC O O P”と呼ぶことにする）と、デバイス・ドライバ用の実行環境（以下、本明細書では“mD r i v e”と呼ぶことにする）はともに、オペレーティング・システムによって提供されているものとする。例えば、表示装置 2 1 の画面上に動画像を表示するためのアプリケーションや、画面表示を介した対話的なユーザ入力を実現するグラフィカル・ユーザ・インターフェース（G U I）

を用いたプログラムなどは、mCOOP上で実行される。また、ディスプレイ・コントローラ13、キーボード22、マウス23、HDDインターフェース16などのハードウェアすなわちデバイスを直接操作する各デバイス・ドライバは、mDrive上で実行される。

【0065】

このようなシステム10において、mCOOP上で動作するアプリケーションとmDrive上で動作するドライバに対しては、同一実行環境上のプログラム間だけでなく、異なる実行環境上のプログラム間でもメッセージ通信を行うことができるようになってきている。例えば、mCOOP上で対話的な画面入力を制御するプログラムは、ディスプレイ・コントローラ13制御用のデバイス・ドライバに対してメッセージ通信を行うことでGUIの実際の描画を実現するが、このデバイス・ドライバは異なる実行環境のmDrive上で動作している。

【0066】

次いで、本実施例に係るオブジェクト指向オペレーティング・システムが提供する実行環境について説明する。

【0067】

図2には、本実施例に係るデータ処理システム10が有するソフトウェア環境を模式的に示している。

【0068】

同図において、オブジェクト2、オブジェクト3、及び、オブジェクト6は、オペレーティング・システムを構成するオブジェクトである。また、オブジェクト2とオブジェクト3は包括されて、1つの複合オブジェクト1を形成している。以下、オブジェクト2及び3を、複合オブジェクト1の「構成オブジェクト」と呼ぶことにする。

【0069】

実行環境7は、複合オブジェクト1やオブジェクト6などオペレーティング・システムを構成する各オブジェクトに対して動作環境を提供する。また、複合オブジェクト1内では、構成オブジェクト2及び3の動作をサポートする独自の実行環境（以下、「構成オブジェクト実行環境」とする）4が用意されている。さ

らに、本実施例では、複合オブジェクト 1 内には、各構成オブジェクトのメッセージ通信履歴に関する情報を保持するコール・ヒストリ 8 と、構成オブジェクトが複合オブジェクト外のオブジェクトへ送出するメッセージ通信を格納するリクエスト・キュー 9 が配設されている。また、実行環境 7 上には、オブジェクト識別子 O I D から参照されるオブジェクト記述情報を保持するオブジェクト・テーブル 5 が用意されている。

【 0 0 7 0 】

複合オブジェクト 1 の外部のオブジェクトからの複合オブジェクト 1 内の構成オブジェクト 2 又は 3 に対する実行要求は、構成オブジェクト実行環境 4 に向けて送信される。

【 0 0 7 1 】

構成オブジェクト実行環境 4 は、該実行要求に応答して、コール・ヒストリ 8 の状態を異なる実行スレッドからの通信状態に更新して、該当する構成オブジェクトを実行するようになっている。

【 0 0 7 2 】

逆に、複合オブジェクト 1 内の構成オブジェクトから複合オブジェクト 1 外のオブジェクトに対するメッセージ送信要求が発行されると、該要求は、同じ複合オブジェクト 1 内にある構成オブジェクト実行環境 4 に送られる。

【 0 0 7 3 】

構成オブジェクト実行環境 4 は、まず、メッセージ受信先のオブジェクトをオブジェクト・テーブル 5 内で検索して、該受信側オブジェクトが同一の複合オブジェクト 1 内に存在するか否かを判断する。

【 0 0 7 4 】

メッセージ受信側が同じ複合オブジェクト 1 内に存在する場合には、構成オブジェクト実行環境 4 は、コール・ヒストリ 8 の状態を同一実行スレッド内からの通信状態に更新して、同一実行スレッド内で該当する構成オブジェクトを実行する。

【 0 0 7 5 】

これに対し、メッセージ受信先が複合オブジェクト 1 の外部のオブジェクトで

あると判断された場合には、構成オブジェクト実行環境 4 は、コール・ヒストリ 8 の状態を異なる実行スレッドからの通信状態に更新するとともに、メッセージ受信先のオブジェクトを示すオブジェクト識別子を付けて、該メッセージをリクエスト・キュー 9 に一旦格納する。

【 0 0 7 6 】

複合オブジェクト 1 は、その実行が終了すると、実行終了要求を構成オブジェクト実行環境 4 に送る。構成オブジェクト実行環境 4 は、コール・ヒストリ 8 の状態を確認して、同一実行スレッドからの通信状態であった場合には、メッセージ送信元である構成オブジェクトの実行を同一スレッドで再開させる。再開させる時点で、コール・ヒストリ 8 を 1 つ前の状態に戻す。これに対し、異なる実行スレッドからの通信状態であることをコール・ヒストリ 8 が示す場合には、構成オブジェクト実行環境 4 は、リクエスト・キュー 9 に格納されている 1 以上のメッセージを 1 回のメッセージ通信でメッセージ送信先に送信して、通常のオブジェクトとしての実行を終了する。

【 0 0 7 7 】

上述したような構成を持つオブジェクト指向オペレーティング・システムによれば、同一複合オブジェクト内における構成オブジェクト間のメッセージ通信のコストが削減されるだけでなく、複合オブジェクト外にある通常のオブジェクトへのメッセージ通信が頻繁に行われる場合であっても、構成オブジェクトからは透過的にメッセージ通信コストを削減することが可能である、という点を充分理解されたい。

【 0 0 7 8 】

なお、図 2 に示すようなソフトウェア構成は、オペレーティング・システムだけではなく、アプリケーションやデバイス・ドライバに対しても同様に適用することができる。

【 0 0 7 9 】

次いで、本実施例に係るオブジェクト指向オペレーティング・システムが提供する異なる実行環境間におけるメッセージ通信サービスについて説明する。

【 0 0 8 0 】

本実施例に係るオブジェクト指向オペレーティング・システムでは、上述したように、異なる実行環境として、アプリケーション用の実行環境mC O O Pと、デバイス・ドライバ用の実行環境mD r i v eを同時に提供する。

【 0 0 8 1 】

図3には、実行環境mC O O P上で動作するアプリケーション51と実行環境mD r i v e上で動作するデバイス・ドライバ52との間でメッセージ通信が行われる様子を示している。

【 0 0 8 2 】

アプリケーション51は、mC O O P上で動作するオブジェクトであり、オペレーティング・システム53が提供するA P I (Application Programming Interface)を利用することができる。アプリケーション51は、メッセージ送信を要求するときには、mC O O Pで提供されているA P I” S e n d W i t h R B o x”を用いる。このとき、S e n d W i t h R B o xへの引数としてメッセージ受信側のオブジェクトを指定するオブジェクト識別子O I Dを付けて、メッセージを渡す。このときの送信先オブジェクトが、実行環境mD i r v e上で動作するデバイス・ドライバ52であったとする。オペレーティング・システム53は、メッセージに付されたO I Dで送信先オブジェクトを特定して、ドライバ52にメッセージを送信するとともに (i n v o k e)、同時にオブジェクト51の実行を再開させる (r e s u m e)。

【 0 0 8 3 】

図4には、メッセージ通信時のオペレーティング・システム53内での動作を詳細に示している。以下、同図を参照しながら、各オブジェクトの相関について説明する。

【 0 0 8 4 】

以下では、オペレーティング・システムを構成するオブジェクトのことを「メタ・オブジェクト」と呼び、アプリケーションやドライバを構成するオブジェクトのことを「ペース・オブジェクト」と呼ぶ。

【 0 0 8 5 】

オペレーティング・システムを構成するメタ・オブジェクトに対しては、mC O

OPやmDriveなどの各ベース・オブジェクト用の実行環境とは相違する、オペレーティング・システム構成用の実行環境が提供されている。本明細書中では、オペレーティング・システム構成用の実行環境のことを、“mCore”と呼ぶことにする。システム実行環境であるmCoreは、メタ・オブジェクトに対して、幾つかのAPI (Application Programming Interface) を提供している。

【0086】

メタ・オブジェクトは2つの方法で起動される。1つは、メタ・オブジェクトに送信されるメッセージによって起動される場合であり、もう1つはベース・オブジェクトが発行するAPIにより起動される場合である。本明細書では、後者のメタ・オブジェクト起動方法のことを特に「メタ・コール」と呼ぶことにする。ベース・オブジェクトの実行環境で提供されるメッセージ通信などのAPIをベース・オブジェクトが使用すると、メタ・コールを発行する。メタ・コールによって、ベース・オブジェクトの実行スレッドから、メタ・オブジェクトの実行スレッドへの遷移が行われる。ベース・オブジェクトとメタ・オブジェクトの動作レベルを切り替えることで、オペレーティング・システムとアプリケーションやドライバとの間に境界を形成して、システム全体の安全性を高めることができる。

【0087】

ここで、異なる実行環境間でメッセージ通信サービスを行う際に必要となるAPIについて説明しておく。

【0088】

Send:

このAPIは、送信メタ・オブジェクトから引数で指定される受信メタ・オブジェクトに対して、引数で指定されるメッセージを送信するためのAPIであり、引数で指定されるメソッドを起動する。受信メタ・オブジェクトがメッセージを受信した後、送信メタ・オブジェクトと受信メタ・オブジェクトは並行に動作する。また、送信メタ・オブジェクトから受信メタ・オブジェクトにメッセージが送信されたときに、受信メタ・オブジェクトが別のメッセージを処理中の場合には、その処理が終了するまで、送信メッセージの処理は開始されず、該メッセージ

はメッセージ・キュー 9 に格納される。

【0089】

Exit :

このAPIは、上記の”Send”APIで起動されたメタ・オブジェクトの実行を終了する。このAPIが発行された時点で、メッセージ・キュー 9 に格納されているメッセージがある場合には、そのメッセージを開始させる。

【0090】

ResumeBase :

このAPIは、メタ・コールにより起動されたメタ・オブジェクトの実行を終了し、メタ・コールを発行したベース・オブジェクトの実行を再開する。これは、メタ・オブジェクトからスケジューラ (Scheduler) 66 へのメタ・コールになる。

【0091】

ExitFromMetaCall :

このAPIは、メタ・コールにより起動されたメタ・オブジェクトの実行を終了する。但し、メタ・コールを発行したベース・オブジェクトの実行は停止したままにする。これは、メタ・オブジェクトからスケジューラ 66 へのメタ・コールになる。

【0092】

なお、mCoreでは、実際には、上述した以外にも数多のAPIを提供されているが、本発明の要旨とは直接関連しないので本明細書中では説明を省略する。また、異なる実行環境間のメッセージ通信サービスを実現するために、本出願人に既に譲渡されている特願平10-283205号明細書に記載されているタグを用いる方法を使用することができる。

【0093】

図4中に示されるmCOOPMailer64は、オペレーティング・システム53を構成するオブジェクトの1つであり、アプリケーション実行環境であるmCOOPに対してメッセージ通信サービスを提供する。本実施例では、mCOOPMailer64は、複合オブジェクトUnifiedMailer57の

構成オブジェクトとして実現されている。

【0094】

mCOOPMailer64は、以下に示すメッセージを受け取り、該当する各動作を実行するようになっている。

【0095】

【表1】

メッセージ	アクション
SendWithRBox	1. RBoxを作成し、RIDを返す。 2. 受信オブジェクトが同一実行環境であれば、受信オブジェクトにメッセージを配送する。 3. 受信オブジェクトが異なる実行環境であれば、その実行環境でメッセージを実現するメタ・オブジェクトに対しDeliverオブジェクトを送信する。
Receive	1. RIDからRBoxを検索する。 2. RBoxに結果メッセージを配送する。
Reply	1. RIDをTIDに変換する。 2. TIDが同一実行環境のものであれば、RIDに対応するRBoxに結果メッセージを配送する。 3. TIDが異なる実行環境のものであれば、DeliverTagメッセージを送信する。
Exit	1. メッセージ・キューにメッセージがあれば、送信する。 2. そうでなければオブジェクトの実行を終了させる。
Deliver	メッセージを配送する。
DeliverTag	結果メッセージを配送する。

【0096】

mCOOPMailer64は、mCOOP上で動作中のベース・オブジェクト（例えば、図4中のアプリケーション52）から送られてくるSendWithRBoxメッセージにより起動して、所定の手続きを実行するようになっている。

【0097】

図5には、mCOOPMailer64がベース・オブジェクトからSendWithRBoxメッセージを受信したことに応答して実行する手続きをフローチャート形式で示している。以下、このフローチャートに従って説明する。

【0098】

まず、ステップS1では、結果メッセージを格納するためのRBoxを作成する。そして、RBoxを識別するための識別子RIDをメッセージ送信元のオブジェクトに返す。

【0099】

次いで、ステップS2では、mCOOPMailer64は、メッセージ受信側のオブジェクトが同一の実行環境上に存在するか否かをチェックする。

【0100】

受信側オブジェクトが同一実行環境上にある場合には、ステップS3に進み、該受信側オブジェクトが別のメッセージを処理中か否かをさらにチェックする。

【0101】

別のメッセージを処理中でないときには、ステップS4に進み、スケジューラ66に対してInvokeメッセージを送信して、メッセージとともに受信オブジェクトを起動する。

【0102】

他方、受信側オブジェクトが別のメッセージを処理中である場合には、ステップS6に進み、メッセージ・キュー9にメッセージを格納する。

【0103】

また、受信オブジェクトが異なる実行環境上にある場合には、ステップS7に進み、その実行環境上でメッセージ通信を実現しているメタ・オブジェクトに対して、Deliverメッセージを送信する。このとき、SendWithRBoxメッセージでベース・オブジェクトが指定したメッセージとRIDをTIDに変換したものも送信する。

【0104】

最後に、"ResumeBase"APIによってベース・オブジェクトの実行を再開させる（ステップS5）。

【0105】

また、mCOOPMailer64は、mCOOPのベース・オブジェクトから送られてきたReceiveメッセージによって起動すると、引数で指定され

たRIDに対応するRBoxを検索する。

【0106】

RBoxに既に結果メッセージが格納されていれば、その結果メッセージをベース・オブジェクトに返し、API"ResumeBase"を発行して、ベース・オブジェクトの実行を再開させる。他方、RBoxに未だ結果メッセージが格納されていなければ、結果メッセージの受け取り領域をRBoxに設定して、API"ExitFromMetaCall"を発行して、ベース・オブジェクトの実行を停止させたまま、mCOOPMailer64の実行を終了する。

【0107】

また、mCOOPMailer64は、mCOOPのベース・オブジェクトから送られるReplyメッセージにより起動すると、API"Reply"を発行した実行スレッドに対応するRIDをTIDに変換して、TIDが同一実行環境上の識別子であるか否か进行检查する。

【0108】

TIDが同一実行環境上のものであった場合には、RIDによりRBoxを検索する。RBoxに対して既に"Receive"APIが発行されていれば、API"Reply"の引数で指定された結果メッセージを"Receive"APIを発行したベース・オブジェクトが指定した領域に格納する。そして、スケジューラ66にResumeメッセージを送信して、"Receive"APIを発行したベース・オブジェクトの実行を再開させる。また、RBoxに対して未だ"Receive"APIが発行されていなければ、"Reply"APIの引数で指定された結果メッセージをRBoxに格納する。

【0109】

他方、TIDが異なる実行環境上のものであった場合には、TIDから結果メッセージを配送する実行環境のメッセージ通信を実現しているメタ・オブジェクトを特定して、そのメタ・オブジェクトに対して、DeliverTagメッセージを送信する。このとき、結果メッセージとTIDを指定する。最後に、"ResumeBase"APIにより、アプリケーションの実行を再開させる。

【0110】

また、mCOOPMailer64は、mCOOPのベース・オブジェクトから送られるExitメッセージにより起動されると、ベース・オブジェクトのメッセージ・キュー9を検査する。

【0111】

メッセージ・キュー9にメッセージが格納されていれば、メッセージ・キューからメッセージを取り出し、スケジューラ66にInvokeメッセージを送信して、メッセージとともに受信オブジェクトを起動する。

【0112】

他方、メッセージ・キュー9にメッセージが格納されていなければ、スケジューラ66にTerminateメッセージを送信して、ベース・オブジェクトの実行を終了させる。

【0113】

また、mCOOPMailer64は、メタ・オブジェクトから送られてくるDeliverメッセージにより起動して、所定の手続きを実行するようになっている。

【0114】

図6には、mCOOPMailer64がメタ・オブジェクトからDeliverメッセージを受信したことに応答して実行する手続きをフローチャート形式で示している。以下、このフローチャートに従って説明する。

【0115】

mCOOPMailer64は、Deliverメッセージで引数に指定された受信オブジェクトに対して、引数で指定されたメッセージを配送する。

【0116】

ステップS11では、受信オブジェクトが別のメッセージに処理中であるか否かを検査する。

【0117】

別のメッセージを処理中でないときには、ステップS12に進み、スケジューラ66に対してInvokeメッセージを送信して、メッセージとともに受信オブジェクトを起動する。

【0118】

他方、受信側オブジェクトが別のメッセージを処理中である場合には、ステップS14に進み、メッセージ・キュー9にメッセージを格納する。

【0119】

最後に、“Exit”APIを発行して、その実行を終了させる（ステップS13）。

【0120】

また、mCOOPMailer64は、メタ・オブジェクトから送られるDeliverTagメッセージにより起動されると、引数で指定されたTIDをRIDに変換して、RIDに対応するRBoxを検索する。

【0121】

RBoxに対して既に“Receive”APIが発行されていれば、“Reply”APIの引数で指定された結果メッセージを“Receive”APIを発行したベース・オブジェクトが指定した領域に格納する。そして、スケジューラ66にResumeメッセージを送信して、“Receive”APIを発行したベース・オブジェクトの実行を再開させる。

【0122】

他方、RBoxに対して未だ“Receive”APIが発行されていないならば、DeliverTagで指定された結果メッセージをRBoxに格納する。mCOOPMailer64は、“Exit”APIによりその実行を終了させる。

【0123】

図4中に示されるmDriveMailer65は、ドライバ実行環境であるmDriveに対してメッセージ通信サービスを提供するオブジェクトであり、mCOOPMailer64と同様に、オペレーティング・システム53を構成するオブジェクトの1つである。本実施例では、mDriveMailer65は、複合オブジェクトUnifiedMailer57の構成オブジェクトとして実現されている。

【0124】

mDriveMailer65は、以下に示すメッセージを受け取り、該当す

る各動作を実行するようになっている。

【0125】

【表 2】

メッセージ	アクション
SendWithContinuation	1. Continuation を作成し、ContIDを返す。 2. 受信オブジェクトが同一実行環境であれば、受信オブジェクトにメッセージを配送する。 3. 受信オブジェクトが異なる実行環境であれば、その実行環境でメッセージを実現するメタ・オブジェクトに対しDeliverオブジェクトを送信する。
Kick	1. ContIDをTIDに変換する。 2. TIDが同一実行環境のものであれば、Continuation に設定された継続オブジェクトに結果メッセージと継続メッセージを配送する。 3. TIDが異なる実行環境のものであれば、DeliverTagメッセージを送信する。
Exit	1. メッセージ・キューにメッセージがあれば、送信する。 2. そうでなければオブジェクトの実行を終了させる。
Deliver	メッセージを配送する。
DeliverTag	結果メッセージを配送する。

【0126】

mDriveMailer65は、mDriveのベース・オブジェクト（例えば、ドライバ52）から送られるSendWithContinuationメッセージによって起動されると、その内部の継続情報を格納するContinuationを作成する。そして、Continuationを識別するための識別子ContIDを、メッセージ送信元であるベース・オブジェクトに返す。また、mDriveMailer65は、メッセージ受信側のオブジェクトが同一の実行環境上に存在するか否か进行检查する。

【0127】

同一実行環境上であった場合には、さらに、該受信側オブジェクトが別のメッセージを処理中か否か进行检查する。別のメッセージを処理中でない場合には、スケジューラ66に対してInvokeメッセージを送信して、メッセージとC o

ntIDとともに受信オブジェクトを起動する。また、受信側のオブジェクトが別のメッセージを処理中であった場合には、メッセージ・キュー9にメッセージを格納する。

【0128】

他方、受信側のオブジェクトが異なる実行環境上に存在する場合には、その実行環境のメッセージ通信を実現しているメタ・オブジェクトに対して、Deliverメッセージを送信する。このとき、SendWithContinuationでベース・オブジェクトが指定したメッセージと、ContIDをTIDに変換したものも送信する。

【0129】

最後に、"ResumeBase"APIによりベース・オブジェクトの実行を再開させる。

【0130】

また、mDriveMailer65は、mDriveのベース・オブジェクトから送られるKickメッセージにより起動されると、"Kick"APIの引数で指定されたContIDをTIDに変換して、TIDが同一実行環境上の識別子であるか否か进行检查する。

【0131】

TIDが同一の実行環境上のものであった場合には、ContIDよりContinuationを検索する。そして、スケジューラ66にInvokeメッセージを送信して、Continuationに設定されている継続メッセージと結果メッセージとともに継続オブジェクトを起動する。

【0132】

他方、TIDが異なる実行環境のものであった場合には、TIDから結果メッセージを配送する実行環境のメッセージ通信を実現しているメタ・オブジェクトを特定して、そのメタ・オブジェクトに対してDeliverTagメッセージを送信する。このとき、結果メッセージとTIDを指定する。

【0133】

最後に、"ResumeBase"APIによりベース・オブジェクトの実行を

再開させる。

【0134】

また、mDriveMailer65は、mDriveのベース・オブジェクトから送られるExitメッセージにより起動されると、ベース・オブジェクトのメッセージ・キュー9を検査する。

【0135】

メッセージ・キュー9にメッセージが格納されていれば、メッセージ・キュー9からメッセージを取り出して、スケジューラ66にInvokeメッセージを送信して、メッセージとともに受信側オブジェクトを起動する。また、メッセージ・キュー9にメッセージが格納されていなければ、スケジューラ66にTerminateメッセージを送信して、ベース・オブジェクトの実行を終了させる。

【0136】

また、mDriveMailer65は、メタ・オブジェクトから送られるDeliverメッセージにより起動されると、図6に示す手続きを実行するようになっている。

【0137】

mDriveMailer65は、Deliverメッセージで引数に指定された受信オブジェクトに対して、引数で指定されたメッセージを配送する。

【0138】

ステップS11では、受信オブジェクトが別のメッセージに処理中であるか否かを検査する。

【0139】

別のメッセージを処理中でないときには、ステップS12に進み、スケジューラ66に対してInvokeメッセージを送信して、メッセージとともに受信オブジェクトを起動する。

【0140】

他方、受信側オブジェクトが別のメッセージを処理中である場合には、ステップS14に進み、メッセージ・キュー9にメッセージを格納する。

【0141】

最後に、"Exit"APIを発行して、その実行を終了させる（ステップS13）。

【0142】

また、mDriveMailer65は、メタ・オブジェクトから送られるDeliverTagメッセージにより起動されると、引数で指定されたTIDをContIDに変換して、ContIDに対応するContinuationを検索する。そして、スケジューラ66にInvokeメッセージを送信して、Continuationに設定されている継続メッセージと結果メッセージとともに継続オブジェクトを起動する。

【0143】

図4中に示されるスケジューラ66は、オブジェクトの実行スレッドのスケジューリングに関するサービスを提供するオブジェクトであり、mCOOPMailer64やmDriveMailer65と同様に、オペレーティング・システム53を構成するオブジェクトの1つである。

【0144】

スケジューラ66は、以下に示すメッセージを受け取り、該当する各動作を実行するようになっている。

【0145】

【表3】

メッセージ	アクション
Invoke	実行スレッドを起動する。
Resume	実行スレッドの実行を再開させる。
Terminate	実行スレッドの実行を終了させる。
ResumeBase	メタ・オブジェクトの実行を終了させ、ベース・オブジェクトの実行を再開させる。
ExitFromMetaCall	メタ・オブジェクトの実行を終了させ、ベース・オブジェクトの実行も停止状態のままにする。

【0146】

スケジューラ66は、メタ・オブジェクトから送られるInvokeメッセージにより起動されると、引数で指定された実行スレッドを、所定のスケジューリ

ング・ポリシーに従ってスケジュールを行い起動する。

【0147】

また、スケジューラ66は、メタ・オブジェクトから送られるResumeメッセージにより起動されると、引数で指定された停止中の実行スレッドの実行を、所定のスケジューリング・ポリシーに従ってスケジュールを行い再開させる。

【0148】

また、スケジューラ66は、メタ・オブジェクトから送られるTerminateメッセージにより起動されると、引数で指定された実行スレッドの実行を終了させる。

【0149】

また、スケジューラ66は、メタ・オブジェクトから送られるResumeBaseメッセージにより起動されると、メタ・オブジェクトの実行を終了させ、メタ・コールを発行したベース・オブジェクトの実行を再開させる。

【0150】

また、スケジューラ66は、メタ・オブジェクトから送られるExitFromMetaCallメッセージにより起動されると、メタ・オブジェクトの実行を終了させ、メタ・コールを発行したベース・オブジェクトも停止状態のままにする。

【0151】

図4中に示されるUnifiedMailer57は、オペレーティング・システム53を構成する複合オブジェクトであり、上述したmCOOPMailer64及びmDriveMailer65をその構成オブジェクトとする。

【0152】

UnifiedMailer57は、その内部の構成オブジェクトに対して、システム環境mCoreと互換性のある"UnifiedMailermCore"APIを提供する。これによって、各構成オブジェクトmCOOPMailer64及びmDriveMailer65を、通常のメタ・オブジェクトであるのと同様にプログラミングすることができる。

【0153】

また、UnifiedMailer57は、"UnifiedMailermCore"APIを実現するために、構成オブジェクトのメッセージ通信履歴情報を保持するためのコール・ヒストリ8を内部に備えている。

【0154】

本実施例では、コール・ヒストリ8は、UnifiedMailer57内の構成オブジェクトがUnifiedMailer57外のオブジェクトからのメッセージ通信やメタ・コールにより起動されたときに、実行スレッドでの実行を示す"External"に設定される。

【0155】

複合オブジェクトUnifiedMailer57がその構成オブジェクトに対して提供する"UnifiedMailermCore"APIについて、以下に説明する。

【0156】

Send:

このAPIは、送信側のメタ・オブジェクトから、引数で指定される受信側のメタ・オブジェクトに対して、引数で指定されるメッセージを送信して、引数で指定されるメソッドを起動する。

【0157】

図7には、"UnifiedMailermCore"APIであるSendの手続きをフローチャートの形式で示している。以下、このフローチャートに従い説明する。

【0158】

まず、ステップS21では、受信側のオブジェクトが複合オブジェクトUnifiedMailer57内の構成オブジェクトであるか否かを検査する。

【0159】

UnifiedMailer57内の構成オブジェクトである場合には、ステップS22に進み、UnifiedMailer57内でメッセージ通信履歴情報を保持するコール・ヒストリ8を、同一実行スレッドでの実行を示すInternalに設定する。

【0160】

次いで、ステップS23では、実行スレッドから受信オブジェクトのエントリを得る。そして、ステップS24では、エントリにジャンプして実行を開始する。このとき、実行スレッドの切り替えを行わず、同一実行スレッドで処理を行う。このため、関数呼び出しと同等の処理が行われる。

【0161】

受信オブジェクトの実行から戻ってきたときには、コール・ヒストリの状態を1つ前の"Send"API呼び出し時の状態に戻して（ステップS25）、"Send"API呼び出し後のポイントから送信オブジェクトの実行を再開する（ステップS26）。

【0162】

また、受信オブジェクトがUnifiedMailer57外のオブジェクトである場合には、ステップS27において、UnifiedMailer57は、その内部に保持するリクエスト・キュー9に、メッセージと受信オブジェクトに関する情報を格納する。

【0163】

Exit:

このAPIは、"Send"APIで起動されたメタ・オブジェクトの実行を終了する。図8には、このExitの手続きをフローチャートの形式で示している。以下、このフローチャートに従い説明する。

【0164】

UnifiedMailer57は、まず、メッセージ通信履歴情報を保持するコール・ヒストリ8の状態を検査する（ステップS31）。

【0165】

コール・ヒストリ8が同一実行スレッドでの実行を示すInternalに設定されている場合には、関数の復帰処理と同等の処理を行い（ステップS32）、呼び出し元の構成オブジェクトに制御を移す。

【0166】

また、コール・ヒストリ8が異なる実行スレッドでの実行を示すExtern

a1で設定されている場合には、UnifiedMailer57が保持するリクエスト・キュー9に格納されているメッセージを受信オブジェクト毎に1つにまとめて（ステップS33）、mCoreの"Send"APIを用いてメッセージの送信を行う。そして、mCoreの"Exit"APIを発行して、メタ・オブジェクトの実行を終了させる（ステップS34）。

【0167】

ResumeBase:

このAPIは、メタ・コールにより起動されたメタ・オブジェクトの実行を終了し、メタ・コールを発行したベース・オブジェクトの実行を再開する。図9には、このResumeBaseの手続きをフローチャートの形式で示している。以下、このフローチャートに従い説明する。

【0168】

UnifiedMailer57は、まず、メッセージ通信履歴情報を保持するコール・ヒストリ8の状態を検査する（ステップS41）。

【0169】

コール・ヒストリ8が同一実行スレッドでの実行を示すInternalに設定されている場合、ステップS42に進み、スケジューラ66を受信オブジェクトに指定して、ResumeBaseメッセージをUnifiedMailer57が保持するリクエスト・キュー9に格納する。そして、リクエスト・キュー9に格納されているメッセージを受信オブジェクト毎に1つにまとめて、mCoreの"Send"APIを用いてメッセージの送信を行う（ステップS43）。スケジューラ66へResumeBaseメッセージが送信されるため、この実行によりメタ・コール処理が終了し、メタ・オブジェクトの実行は終了して、同時にベース・オブジェクトの実行が再開する。

【0170】

他方、コール・ヒストリ8が異なる実行スレッドでの実行を示すExternalに設定されている場合、ステップS44に進んで、エラーを発生させる。

【0171】

ExitFromMetaCall:

このAPIは、図9に示したものと同様の手順に従い、メタ・コールにより起動されたメタ・オブジェクトの実行を終了する。但し、メタ・コールを発行したベース・オブジェクトの実行は停止したままにする。

【0172】

UnifiedMailer57は、まず、メッセージ通信履歴情報を保持するコール・ヒストリ8の状態を検査する（ステップS41）。

【0173】

コール・ヒストリ8が同一実行スレッドでの実行を示すInternalに設定されている場合、ステップS42では、ResumeBaseメッセージとなっているが、その代わりにExitFromMetaCallメッセージを、スケジューラ66を受信オブジェクトに指定してUnifiedMailer57が保持するリクエスト・キュー9に格納する。そして、ステップS43では、リクエスト・キュー9に格納されているメッセージを受信オブジェクト毎に1つにまとめて、mCCoreの"Send"APIを用いてメッセージの送信を行う。スケジューラ66へはExitFromMetaCallメッセージが送信されるため、この実行によりメタ・コール処理が終了し、メタ・オブジェクトの実行は終了して、同時にベース・オブジェクトの実行もメタ・コール発行時点で停止したままになる。

【0174】

他方、コール・ヒストリ8が異なる実行スレッドでの実行を示すExternalに設定されている場合、ステップS44に進んで、エラーを発生させる。

【0175】

次いで、本実施例に係るオブジェクト指向オペレーティング・システムが提供する、異なる実行環境間でメッセージ通信を行う場合のシナリオについて説明する。

【0176】

図10には、実行環境mCOOP上で動作するアプリケーションから他の実行環境mDrive上で動作するドライバに対して、mCOOPにより提供される"SendWithRBox"APIを用いてメッセージ通信を行う場合の各オブ

ジェクト間の処理手順を示している。但し、図面の簡素化のため、メッセージ返信の処理については省略している。

【0177】

アプリケーションは、mCOOPで提供される"SendWithRBox"APIを処理P1で発行する。このとき、"SendWithRBox"APIはメタ・コールを発行するため、ベース・オブジェクトの実行スレッドからメタ・オブジェクトの実行スレッドへの遷移が行われる。また、複合オブジェクトUnifiedMailer57外からのメタ・コールであるため、UnifiedMailer57におけるメッセージ通信履歴情報を保持するコール・ヒストリ8は、外部の実行スレッドからの実行であることを示すExternalに設定される。

【0178】

mCOOPMailer64がSendWithRBoxメッセージを受け取ると、次の処理P2において、図5に示す手続きを行う。mCOOPMailer64は、まず、ステップS1では、結果メッセージを格納するためのRBoxを作成する。そして、RBoxを識別するためのRID識別子をメッセージ送信オブジェクトのRID格納領域に設定し、"SendWithRBox"APIの終了時にアプリケーションがRIDを得られるようにする。RID格納領域は、"SendWithRBox"APIの引数として、アプリケーションにより与えられる。

【0179】

次いで、ステップS2では、メッセージ受信側のオブジェクトが同一の実行環境上に存在するか否かを検査する。これは、OIDを用いてオブジェクト・テーブル70から得られるオブジェクト記述情報を検索することで行われる。

【0180】

図10に示すシナリオでは、受信側オブジェクトであるドライバは実行環境mDrive上のオブジェクトであるので、ステップS2では、送信側の実行環境mCOOPとは異なる実行環境上のオブジェクトとして判断される。このため、判断ブロックの分岐NoからステップS7に進む。

【0181】

ステップS7では、mCOOPMailerは、受信側のオブジェクトの実行環境上でメッセージ通信を実現するメタ・オブジェクトに対して、UnifiedMailermCoreのAPIである"Send"を用いてDeliverメッセージを送信する。

【0182】

図10に示すシナリオでは、mDriveMailer65に対してSendを発行することになる。Deliverメッセージには、"SendWithRBox"APIの引数で指定されたメッセージとRIDをTIDに変換したものを付加する。

【0183】

複合オブジェクトUnifiedMailer57が提供するUnifiedMailerCoreの"Send"APIは、図7に示す手続きを行う。

【0184】

まず、ステップS21では、受信側のオブジェクトが複合オブジェクトUnifiedMailer57内の構成オブジェクトであるか否かを検査する。この検査は、OIDを用いてオブジェクト・テーブル70を検索して得られるオブジェクト記述情報を使うことで行われる。

【0185】

図10に示すシナリオでは、受信側のオブジェクトmDriveMailer65は、同一の複合オブジェクトUnifiedMailer57内の構成オブジェクトであるので、次ステップS22に進む。

【0186】

ステップS22では、UnifiedMailer57内でメッセージ通信履歴情報を保持するコール・ヒストリ8を、同一実行スレッドでの実行を示すInternalに設定する。

【0187】

次いで、ステップS23では、実行スレッドから受信オブジェクトのエントリを得る。そして、ステップS24では、エントリにジャンプして実行を開始する

。図10中において、遷移M2として示される点線矢印がこの遷移に該当する。ここで、遷移M2を点線で示しているのは、実際には実行スレッドの切り替えが行われていないことを意味している。

【0188】

mDriveMailer65がDeliverメッセージを受け取ると、図10の処理P3では図6に示す手続きが行われる。

【0189】

ステップS11では、受信オブジェクトが別のメッセージに処理中であるか否かを検査する。この検査は、受信オブジェクトのようなメッセージ・キュー9が空であるか否かで判断することができる。そして、現在メッセージの処理を行っていないければ、mDriveMailer65は、ステップS12に進んで、UnifiedMailermCoreの"Send"APIを用いて、スケジューラ66に対してInvokeメッセージを送信する。

【0190】

UnifiedMailermCoreが提供する"Send"APIは、図7に示す手続きを行う。

【0191】

まず、ステップS21では、受信側のオブジェクトが複合オブジェクトUnifiedMailer57内の構成オブジェクトであるか否かを検査する。

【0192】

図10に示すシナリオでは、受信オブジェクトとしてのスケジューラ66は、複合オブジェクトUnifiedMailer57外の通常のオブジェクトであるので、ステップS27に進む。

【0193】

ステップS27では、受信オブジェクトを示すOIDとメッセージがUnifiedMailer57内のリクエスト・キュー9に格納される。

【0194】

図10に示す遷移M3及びM4はいずれも点線矢印で示されているが、これらは、実際には実行スレッドの切り替えが行われていないことを意味する。すなわ

ち、この時点では、複合オブジェクト57内の構成オブジェクトの観点からはスケジューラ66へのメッセージ送信が実行されたことになるが、実際には、メッセージ通信は透過的に遅延される。したがって、図10の処理P4は、この時点では未だ処理が行われない。

【0195】

mDriveMailer65は、図10中の処理P3でのUnifiedMailermCoreの"Send"APIから戻ると、処理P5において、Deliverメッセージの処理の続きを行う。すなわち、図6のステップS13において、UnifiedMailermCoreの"Exit"APIを発行する。

【0196】

UnifiedMailermCoreが提供する"Exit"APIは、図8に示す手続きを行う。

【0197】

UnifiedMailer57は、まず、メッセージ通信履歴情報を保持するコール・ヒストリ8の状態を検査する（ステップS31）。

【0198】

図10に示すシナリオでは、mCOOPMailer64からmDriveMailer65に対してUnifiedMailermCoreの"Send"APIを発行したときに、図7のステップS22においてInternalが設定されている。したがって、UnifiedMailermCoreの"Exit"APIは、図8のステップS32に進む。

【0199】

ステップS32では、UnifiedMailermCoreの"Send"APIの発行元に実行を復帰させる。これは、関数の復帰と同様の処理で行われる。復帰処理により、mCOOPMailer64は、図10の処理P2におけるUnifiedMailermCoreの"Send"APIの呼び出し手続きの続きを行う。

【0200】

図10に示す遷移M5は点線矢印で示されているが、これらは、実際には実行スレッドの切り替えが行われていないことを意味する。

【0201】

この結果、図7のステップS25に実行が移る。ステップS25では、コール・ヒストリの状態を1つ前の状態に戻す。図10に示すシナリオの場合、コール・ヒストリ8の状態は、mCOOPMailer64からmDriveMailer65に対してUnifiedMailerCoreの"Send"APIを発行したときに設定されたInternal状態（図7のステップS22を参照のこと）から、アプリケーションがメタ・コールとしてmCOOPの"SendWithRBox"APIを発行してUnifiedMailer57に実行スレッドが切り替わったときに設定されたExternal状態に戻る。すなわち、図5のステップS5において、UnifiedMailerCoreの"ResumeBase"APIを発行する。

【0202】

mCOOPMailer64が提供するUnifiedMailerCoreの"ResumeBase"APIは、図9に示す手続きを行う。

【0203】

UnifiedMailer57は、まず、メッセージ通信履歴情報を保持するコール・ヒストリ8の状態を検査する（ステップS41）。

【0204】

図10に示すシナリオでは、アプリケーションがmCOOPの"SendWithRBox"APIを発行した時点の状態、すなわち、メタ・コールとしてUnifiedMailer57に実行スレッドが切り替わったときに設定されたExternalに状態が戻されているため、UnifiedMailerCoreの"ResumeBase"APIはステップS42に進む。

【0205】

ステップS42では、スケジューラ66のOIDとスケジューラ66へのResumeBaseメッセージをUnifiedMailer57内のリクエスト・キュー9に格納する。そして、ステップS43では、UnifiedMail

er 57内のリクエスト・キュー9に格納されているメッセージを受信オブジェクト毎に1つにまとめて、mCoreの"Send"APIを用いてメッセージの送信を行う。

【0206】

図10に示すシナリオでは、遷移M3で送信されず遅延されていたスケジューラ66へのInvokeメッセージと、図9のステップS42で格納したスケジューラ66へのResumeBaseメッセージが1つにまとめられて、1回のmCoreの"Send"APIを用いてスケジューラ66に送られることになる。

【0207】

スケジューラ66は、InvokeとResumeBaseの2つのメッセージを受け取ると、図10の処理P7において、本来は処理P4において行われるはずが遅延されていたInvokeと、新たなResumeBaseの処理を行う。

【0208】

スケジューラ66は、Invoke処理を受けてスケジューリングを行い、遷移M7で通信先オブジェクトとなるドライバの実行スレッドを起動する。これによって、実行環境mDrive上で動作するドライバは、異なる実行環境mCOOP上で動作するアプリケーションからmCOOPのAPI"SendWithRBox"で送られたメッセージを受け取り、該メッセージを処理P8で処理を行うことができる。

【0209】

また、スケジューラ66は、ResumeBaseの処理を行い、UnifiedMailer 57内のメタ・オブジェクトの実行を終了させるとともに、図10には示していないが、スケジューリング結果の適切な時点でアプリケーションの実行を再開させる。この結果、アプリケーションは、mCOOPが提供するAPI"SendWithRBox"のメタ・コール呼び出しから復帰して、以後の処理を継続することができる。

【0210】

以上を総括すれば、メッセージ通信が頻繁に発生するような複数のオブジェクトを複合オブジェクトとして包括するデータ通信方式に対して、さらに、複合オブジェクト外へのメッセージ通信を一時的に格納するリクエスト・キュー9と構成オブジェクト間のメッセージ通信履歴情報を保持するコール・ヒストリ8を追加する。そして、構成オブジェクトmCOOPMailer及びmDriveMailerが複合オブジェクトUnifiedMailer外のオブジェクトであるスケジューラ66に対してメッセージ通信を行うと、該メッセージを、受信オブジェクトとしてのスケジューラを示すOIDとともにリクエスト・キュー9に一旦格納してメッセージ通信を遅延させる。その後、構成オブジェクトの実行終了時にコール・ヒストリの状態の検査を行い、それが外部の実行スレッドからの実行を示すExternalであった場合には、リクエスト・キュー9に格納された複数のメッセージを各受信オブジェクト毎に1つにまとめて、1回のメッセージ通信で送信を行うことができる。つまり、各構成オブジェクトmCOOPMailer及びmDriveMailer間でメッセージ通信を行っている期間中は、外部のオブジェクトへのメッセージ通信を遅延させる。そして、複合オブジェクトUnifiedMailerとしての動作が終了したときに、複数のメッセージをまとめて送信することで、メタ・オブジェクト間のメッセージ通信回数を削減し、mCOOPとmDriveという異なる実行環境間でのメッセージ通信サービスの性能を向上させることができるという訳である。

【0211】

本発明によれば、複合オブジェクトUnifiedMailerの各構成オブジェクトmCOOPMailer及びmDriveMailer間でメッセージ通信を行う際に、複合オブジェクト外のオブジェクトであるスケジューラに対するメッセージが発生するにも拘わらず、コストを削減することができる。図10に示す例では、遷移M2及びM5の間に挿入された遷移M3及びM4を、複合オブジェクト内の構成オブジェクトからは透過的に削除することができる。通常のオブジェクトのプログラミングを全く変更せずに、複合オブジェクトUnifiedMailerに構成オブジェクトとしてリクエスト・キューやコール・ヒストリを追加するだけで実装することができる。

【0212】

すなわち、本発明によれば、メッセージ通信が頻繁に発生するような複数のオブジェクトを複合オブジェクトとして包括するデータ通信方式によって、メッセージ通信のコストを削減することができる訳である。特に、複合オブジェクト内の異なる構成オブジェクトで発行されたメッセージ通信であっても、それを遅延させて1つのメッセージ通信として送信することができる。

【0213】

[追補]

以上、特定の実施例を参照しながら、本発明について詳解してきた。しかしながら、本発明の要旨を逸脱しない範囲で当業者が該実施例の修正や代用を成し得ることは自明である。すなわち、例示という形態で本発明を開示してきたのであり、限定的に解釈されるべきではない。本発明の要旨を判断するためには、冒頭に記載した特許請求の範囲の欄を参酌すべきである。

【0214】

【発明の効果】

以上詳記したように、本発明によれば、複数のオブジェクト間で好適にメッセージ交換を行うことができる、優れたデータ通信方式を提供することができる。

【0215】

また、本発明によれば、複数の並列オブジェクトで構成されるオブジェクト指向オペレーティング・システムにおいてオブジェクト間で好適にメッセージ交換を行うことができる、優れたデータ通信方式を提供することができる。

【0216】

また、本発明によれば、コンテキスト・スイッチなどのコスト発生を可能な限り抑制して、オブジェクト間のメッセージ交換を高速化することができる、優れたデータ通信方式を提供することができる。

【0217】

また、本発明によれば、オブジェクト指向オペレーティング・システムにおけるシステム構成の柔軟性や動的変更の容易性などの優位性を維持したままで、オブジェクト間のメッセージ通信時のコストを削減して、システム全体の実行性能

を向上させることができる。

【0218】

また、本発明によれば、複合オブジェクト内のオブジェクトと複合オブジェクト外に存在するオブジェクトとのメッセージ通信が頻繁に行われている場合、複合オブジェクト内の各オブジェクトには透過的な形式でメッセージが蓄積され、その後の適切なタイミングで蓄積された複数のメッセージをまとめて1回のメッセージ通信による送信することができる。この結果、メッセージ通信のコストを大幅に削減することができる。また、オブジェクトの独立性を保持したままシステム全体の実行性能を向上させることができる。

【図面の簡単な説明】

【図1】

本発明の実施に供されるデータ処理システム10のハードウェア構成を模式的に示した図である。

【図2】

本実施例に係るデータ処理システム10が有するソフトウェア環境を模式的に示した図である。

【図3】

オブジェクト指向オペレーティング・システムが提供する異なる実行環境間で行われるメッセージ通信の動作を示した図であり、より具体的には、実行環境mCOOP上で動作するアプリケーションと実行環境mDrive上で動作するデバイス・ドライバとの間でメッセージ通信が行われる様子を示した図である。

【図4】

実行環境mCOOP上で動作するアプリケーションと実行環境mDrive上で動作するデバイス・ドライバとの間でメッセージ通信が行われる様子を示した図であり、より具体的には、メッセージ通信時のオペレーティング・システム内での動作を詳細に示した図である。

【図5】

メタ・オブジェクトmCOOPMailer64がベース・オブジェクトからSendWithRBoxメッセージを受信したことに応答して実行する手続きを

示したフローチャートである。

【図6】

メタ・オブジェクトmCOOPMailer64又はmDriveMailer65がベース・オブジェクトからDeliverメッセージを受信したことに応答して実行する手続きを示したフローチャートである。

【図7】

"UnifiedMailermCore"APIであるSendの手続きを示したフローチャートである。

【図8】

"UnifiedMailermCore"APIであるExitの手続きを示したフローチャートである。

【図9】

"UnifiedMailermCore"APIであるResumeBase及びExitFromMetaCallの手続きを示したフローチャートである。

【図10】

実行環境mCOOP上で動作するアプリケーションから他の実行環境mDrive上で動作するドライバに対して、mCOOPに提供される"SendWithRBox"APIを用いてメッセージ通信を行う場合の各オブジェクト間の処理手順を示した図である。

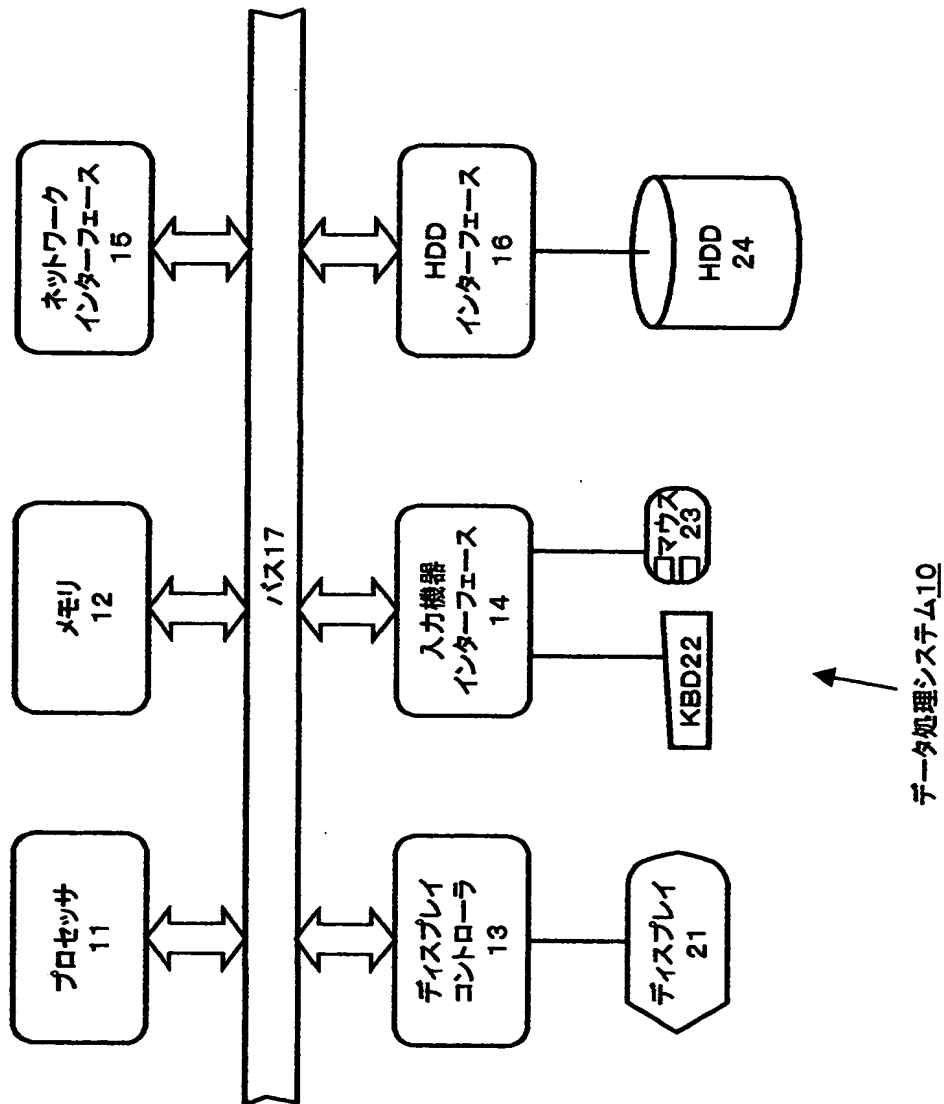
【符号の説明】

- 1…複合オブジェクト
- 2, 3…構成オブジェクト
- 4…構成オブジェクト実行環境
- 5…オブジェクト・テーブル
- 6…オブジェクト
- 7…実行環境
- 8…コール・ヒストリ
- 9…リクエスト・キュー

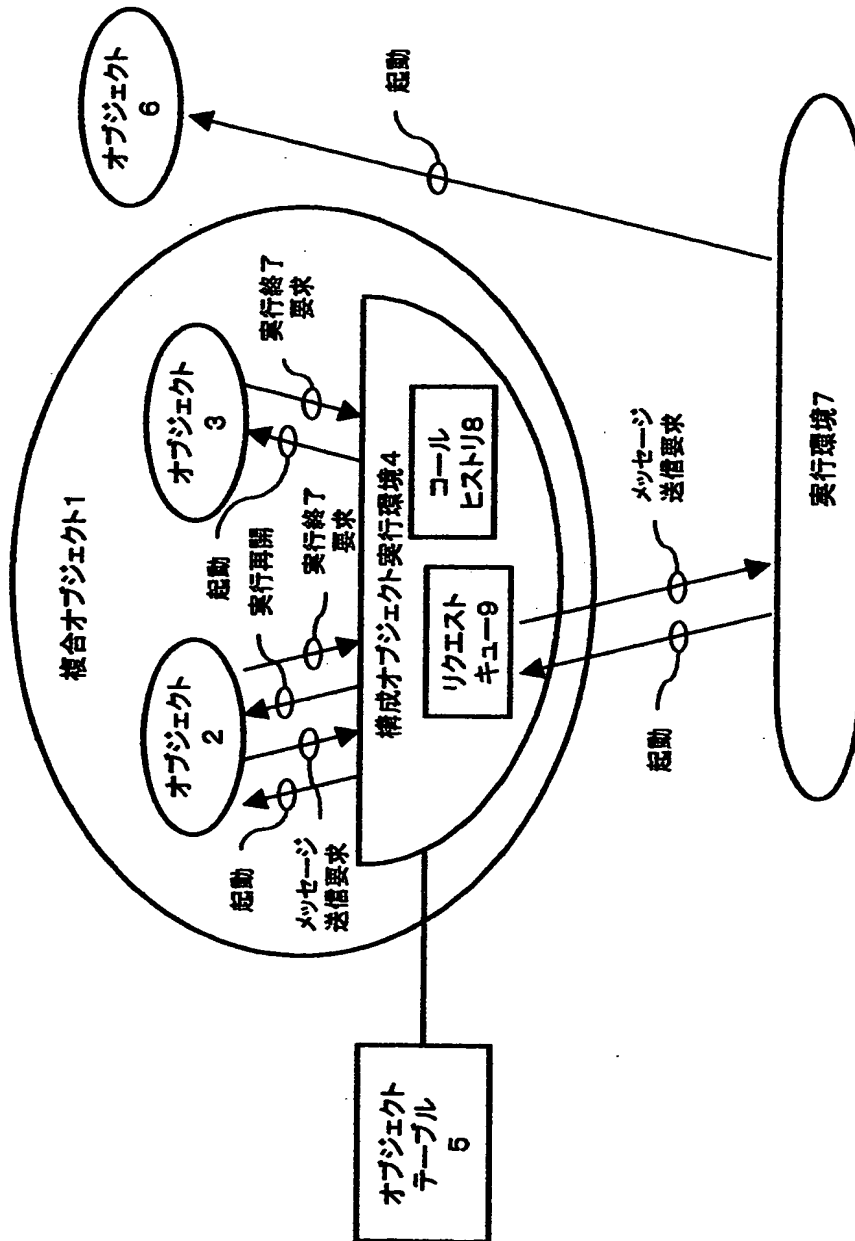
- 10…データ処理システム
- 11…プロセッサ
- 12…メモリ
- 13…ディスプレイ・コントローラ
- 14…入力機器インターフェース
- 15…ネットワーク・インターフェース
- 16…HDDインターフェース
- 21…ディスプレイ
- 22…キーボード
- 23…マウス
- 24…HDD
- 51…アプリケーション
- 52…ドライバ
- 53…オペレーティング・システム
- 64…mCOOPMailer
- 65…mDriveMailer
- 66…スケジューラ

【書類名】 図面

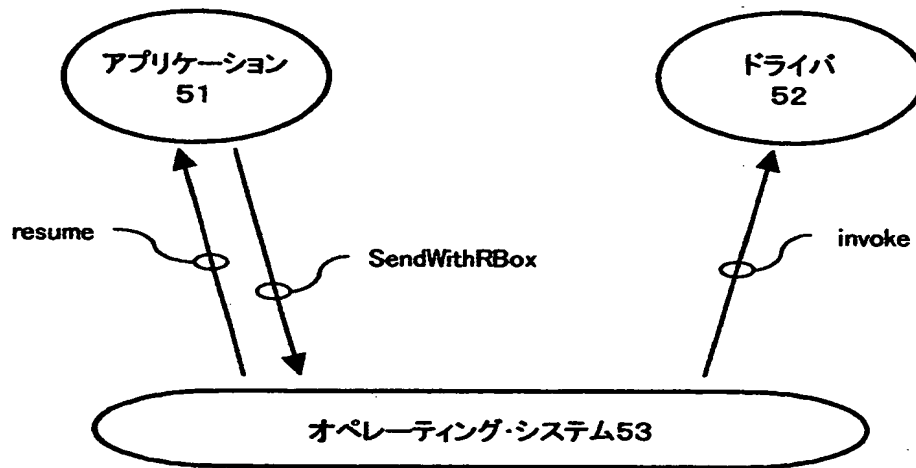
【図 1】



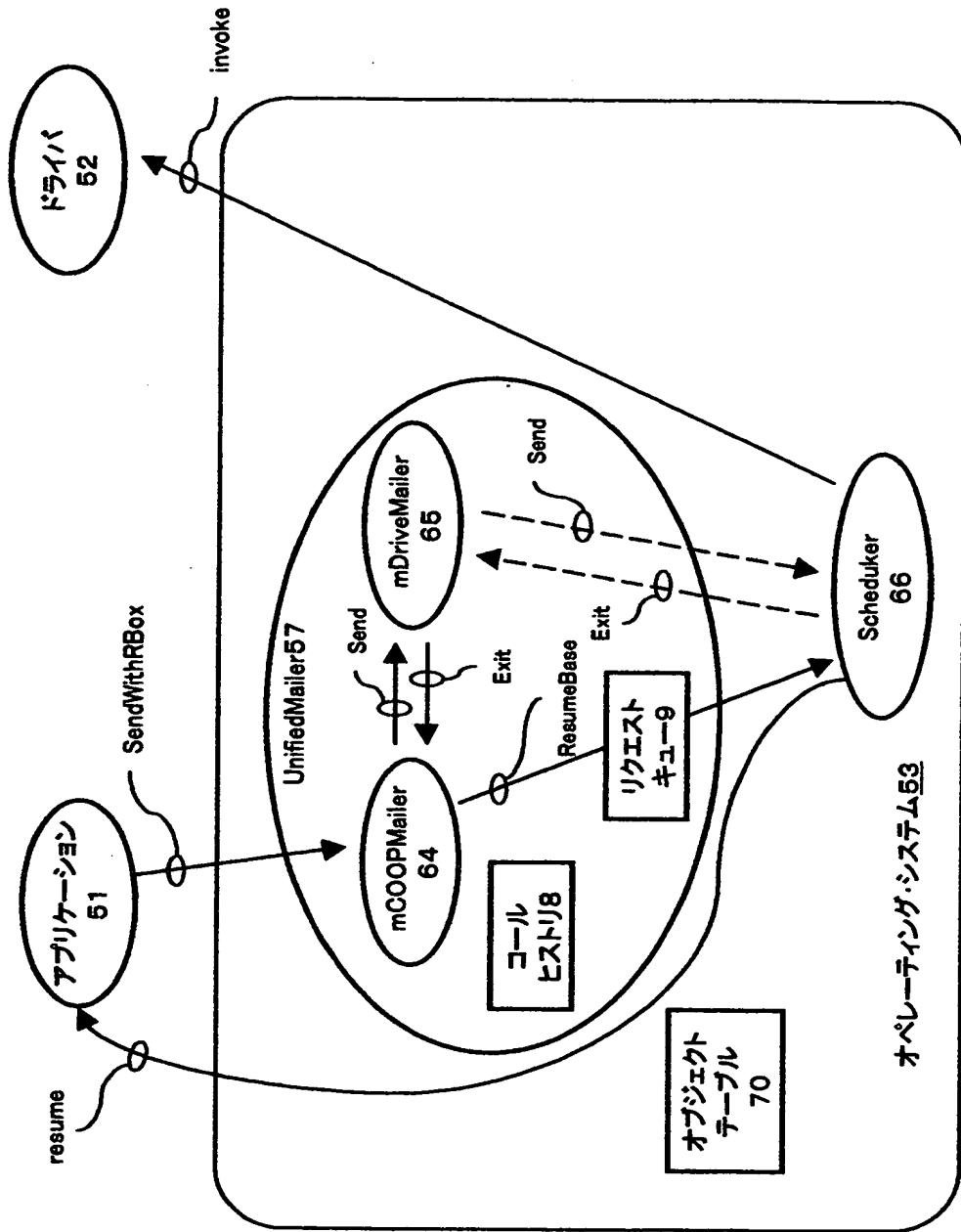
【図2】



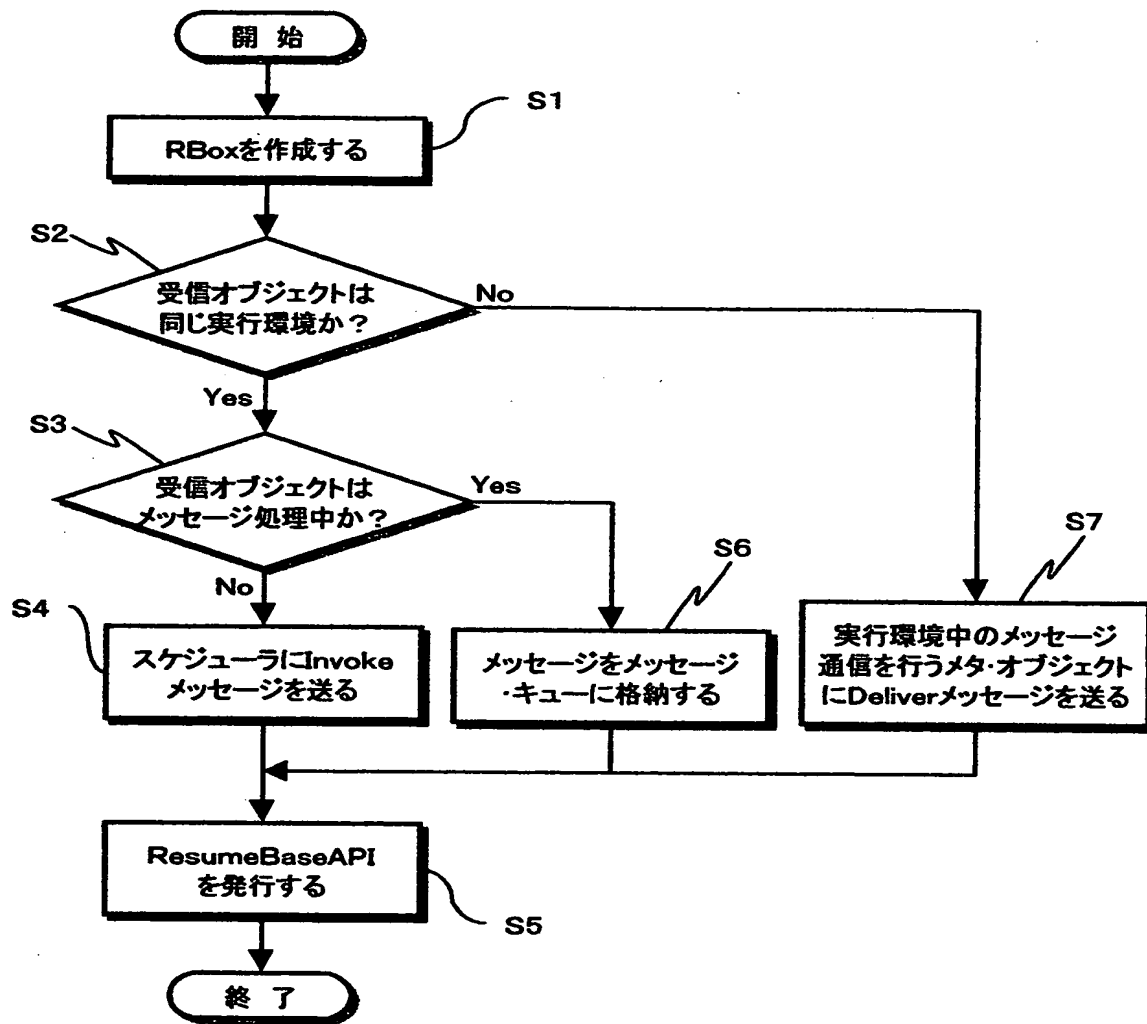
【図3】



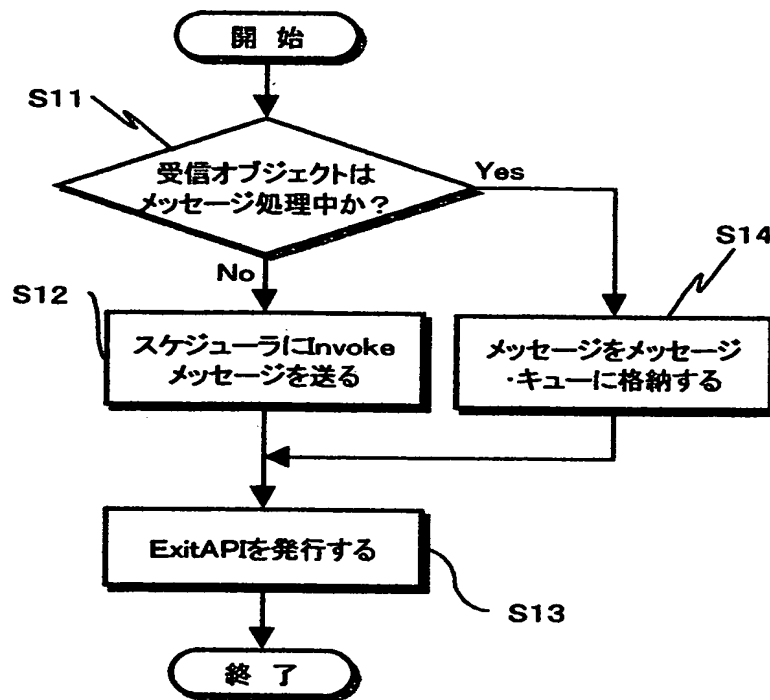
【図4】



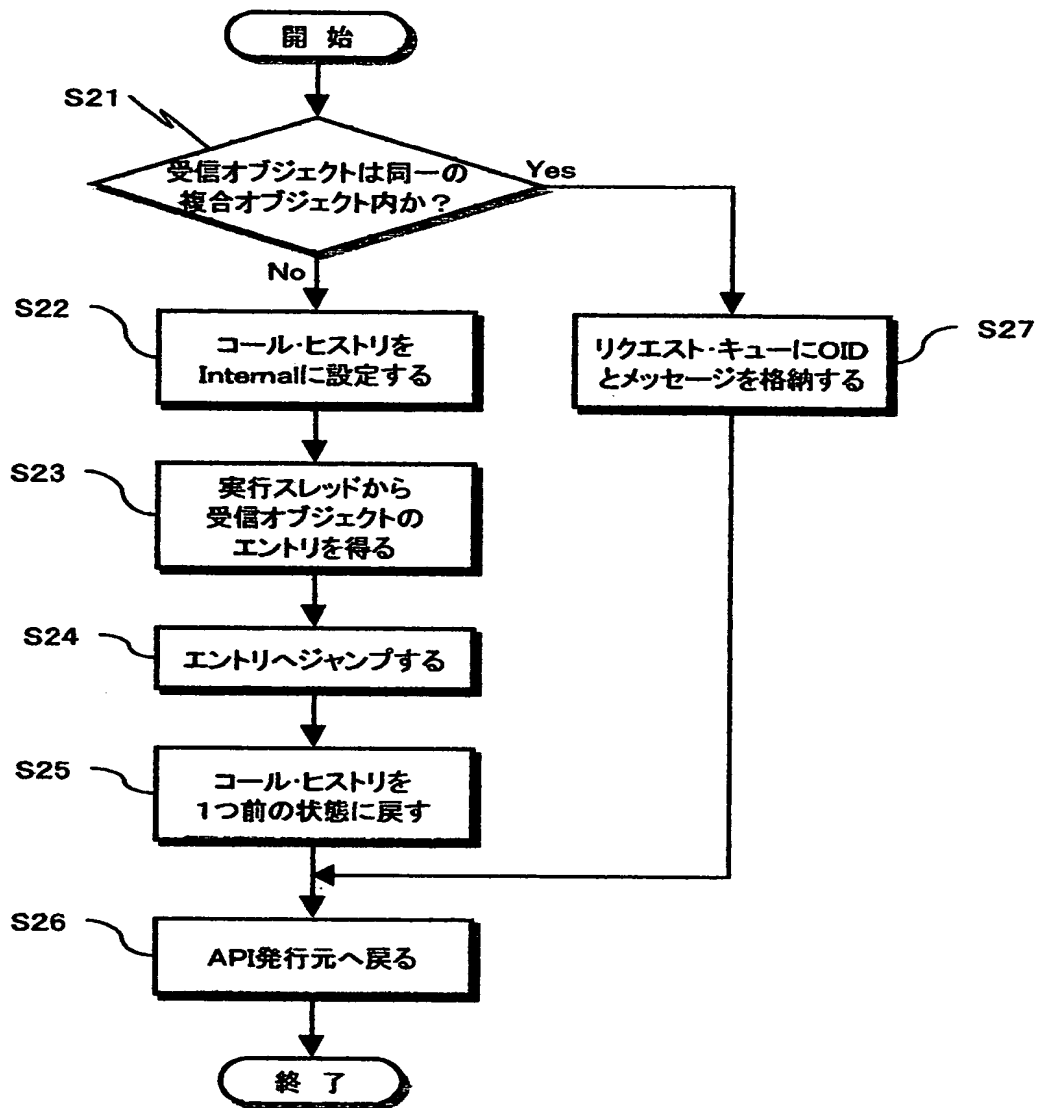
【図5】



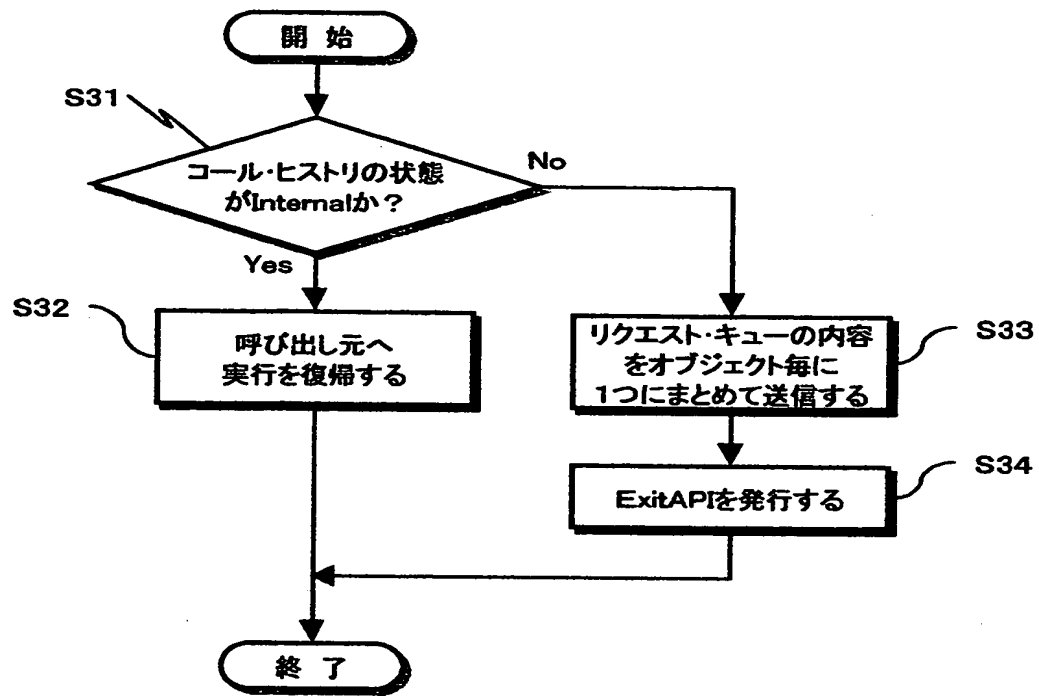
【図6】



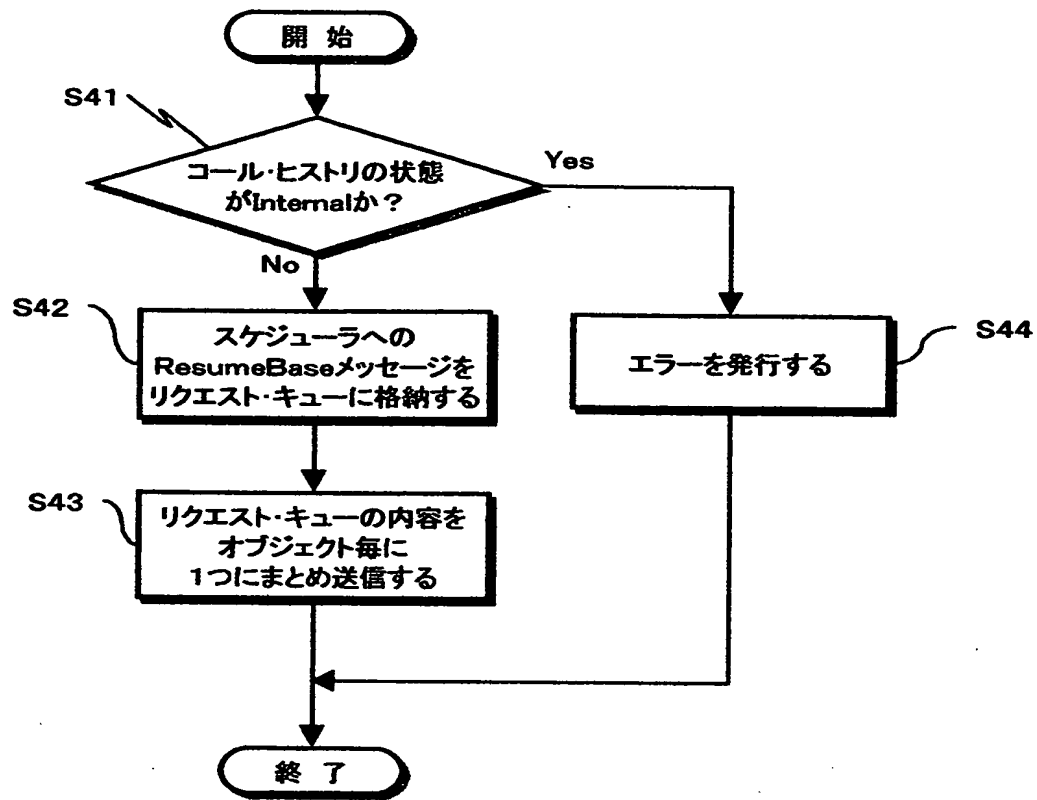
【図 7】



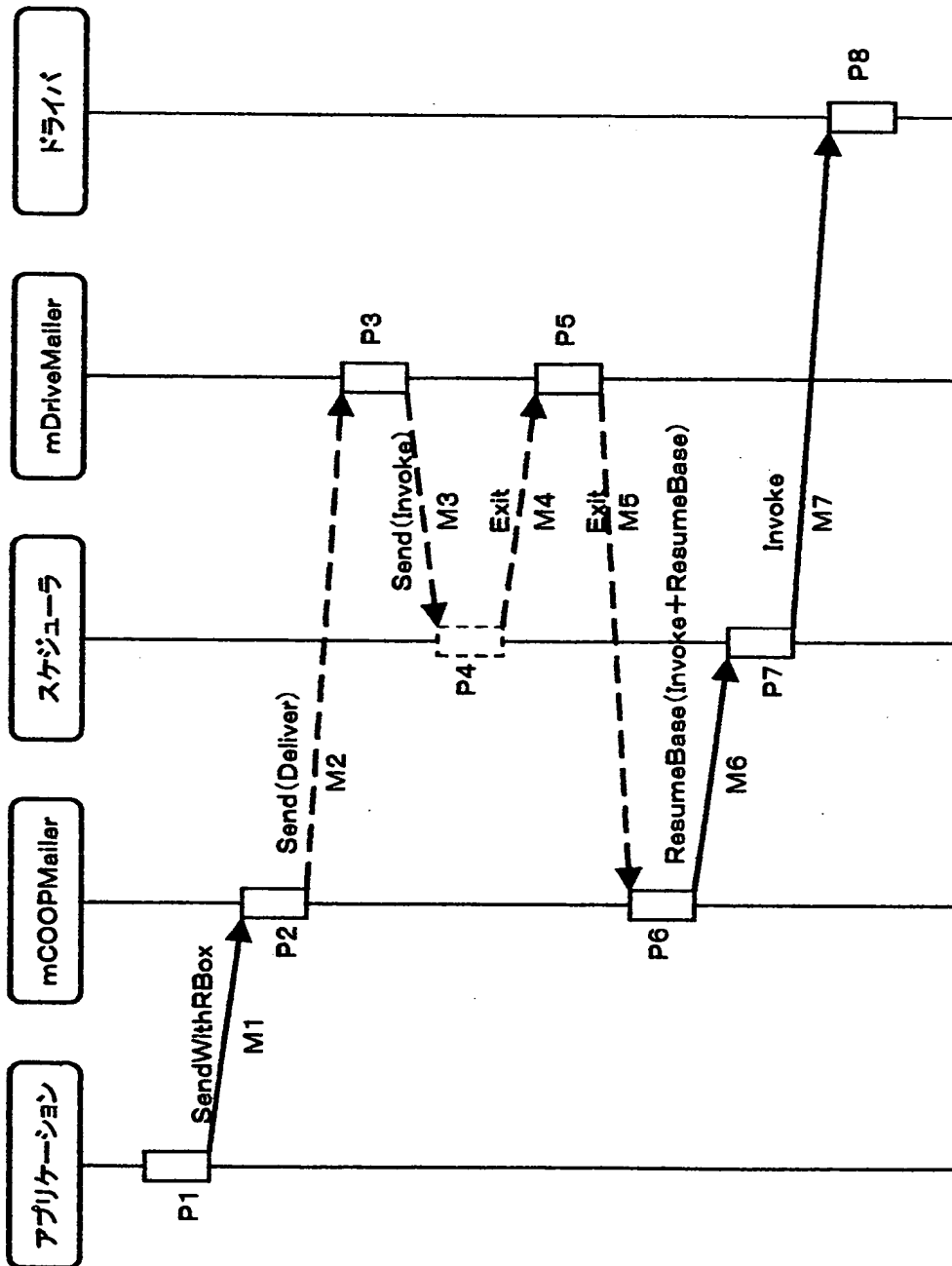
【図 8】



【図9】



【図 10】



【書類名】 要約書

【要約】

【課題】 オブジェクト指向オペレーティング・システムにおけるシステム構成の柔軟性などの優位性を維持したままで、メッセージ通信時のコストを削減する。

【解決手段】 複合オブジェクト内でのメッセージ通信のときは、同一の実行スレッドで実行して、メッセージ通信履歴を同一実行スレッドからの通信状態に更新する。複合オブジェクト外へのメッセージ送信のときは、メッセージ送信先に応じてメッセージを一旦蓄積する。複合オブジェクト内でのメッセージ処理が終了後にメッセージ通信履歴を参照して、異なる実行スレッドからの通信状態であれば、蓄積されたメッセージを1回のメッセージ通信で送信する。

【選択図】 図4

出 願 人 履 歴 情 報

識別番号 [000002185]

1. 変更年月日	1990年 8月30日
[変更理由]	新規登録
住 所	東京都品川区北品川6丁目7番35号
氏 名	ソニー株式会社